INSTITUTO MILITAR DE ENGENHARIA

RICARDO MAROQUIO BERNARDO

- SIMUBLIMP – UMA CONTRIBUIÇÃO AO DESENVOLVIMENTO DE ALGORITMOS INTELIGENTES PARA UMA EQUIPE DE DIRIGÍVEIS ROBÓTICOS AUTÔNOMOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.
Orientador: Prof. Paulo Fernando Ferreira Rosa, Ph.D.

Rio de Janeiro
2007

c2007

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro - RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do autor e do orientador.

BS181 Bernardo, Ricardo Maroquio.
SimuBlimp: uma Contribuição ao Desenvolvimento de Algoritmos Inteligentes para Uma Equipe de Dirigíveis Robóticos Autônomos / Ricardo Maroquio Bernardo. – Rio de Janeiro: Instituto Militar de Engenharia, 2007.
176 p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2007

1. VANT. 2. Simulação. 3. Dirigível. I. SimuBlimp: uma Contribuição ao Desenvolvimento de Algoritmos Inteligentes para Uma Equipe de Dirigíveis Robóticos Autônomos. II. Instituto Militar de Engenharia

CDD 005.1

INSTITUTO MILITAR DE ENGENHARIA

RICARDO MAROQUIO BERNARDO

SIMUBLIMP – UMA CONTRIBUIÇÃO AO DESENVOLVIMENTO DE ALGORITMOS INTELIGENTES PARA UMA EQUIPE DE DIRIGÍVEIS ROBÓTICOS AUTÔNOMOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Paulo F. F. Rosa, Ph.D.

Aprovada em 16 de julho de 2007 pela seguinte Banca Examinadora:

_____

Prof. Paulo Fernando Ferreira Rosa – Ph.D. do IME – Presidente

_____

Prof. Esteban Walter Gonzalez Clua – D.Sc. da UFF

_____

Prof. Paulo César Pellanda – Dr. ENSAE do IME

_____

Prof. Antonio Eduardo Carrilho da Cunha – D.Sc. do IME

Rio de Janeiro
2007

AGRADECIMENTOS

Primeiramente a Deus, que sempre me amparou nos momentos de aflição e angústia. Aos meus pais, João Bernardo e Maria de Fátima, e à minha avó, Dozolina Zanetti, pelos valores que me transmitiram e pelo apoio espontâneo que me deram ao longo da vida, sempre. À minha esposa, Polyana, pela paciência e compreensão à minha constante ausência durante o desenvolvimento desse trabalho.

Aos meus professores e orientadores, de mestrado, Paulo Fernando Ferreira Rosa, e de graduação, Raul Fonseca Neto, por serem mais do que apenas professores e por terem me motivado a "mergulhar" no fantástico mundo da pesquisa.

Aos amigos e amigas de mestrado, Fábio Vidal, Monael Pinheiro, Liliana Paola, Marlos de Mendonça, Marco Antônio Firmino de Sousa, Rafael Lima de Carvalho e outros (cujos nomes não caberiam neste espaço), pelas palavras de incentivo e pelas inúmeras vezes que me representaram em minha ausência.

Aos amigos das antigas, Sérgio Freitas, Diogo Mattos, Pablo Mendes, Kele Teixeira, Viviane Kawamura, Flávio Flores, Saulo Vilella e demais membros da maravilhosa turma de graduação da UFJF que, assim como eu, foram encarar um mestrado no Rio de Janeiro, o que acabou resultando em um reencontro excepcional.

Ao Instituto Militar de Engenharia, pela oportunidade que me deu de fazer o curso de mestrado em Sistemas e Computação, o qual acarretou o desenvolvimento do presente trabalho. Aos membros que compuseram a banca durante defesa desta dissertação, pelas ótimas sugestões e conselhos, fundamentais para o fechamento do trabalho.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pela ajuda financeira, através de bolsa de estudo, sem a qual o desenvolvimento desse trabalho teria sido muito mais árduo, ou, possivelmente, inviável.

A todos aqueles que me apoiaram, direta ou indiretamente, cujos nomes não caberiam neste documento.

Ricardo Maroquio Bernardo

SUMÁRIO

LISTA DE ILUSTRAÇÕES 9
LISTA DE TABELAS 11
LISTA DE SIGLAS 12

1 INTRODUÇÃO 16
1.1 HISTÓRICO E VISÃO GERAL 16
1.2 MOTIVAÇÃO 20
1.3 OBJETIVOS E CONTRIBUIÇÕES 23
1.4 ESTRUTURA DA DISSERTAÇÃO 25

2 REVISÃO DE LITERATURA 27
2.1 TRABALHOS RELACIONADOS 27
2.2 TRABALHOS ANTERIORES 32

3 FUNDAMENTAÇÃO TEÓRICA 40
3.1 AGENTES, AMBIENTES E SISTEMAS MULTIAGENTES 40
3.1.1 AGENTES 40
3.1.2 AMBIENTES 42
3.1.3 SISTEMA MULTIAGENTES 44
3.2 ROBÓTICA 47
3.2.1 SENSORES 49
3.2.2 EFETUADORES 53
3.2.3 OUTROS HARDWARES DE ROBÔS 55
3.2.4 PERCEPÇÃO ROBÓTICA 55
3.3 COMPUTAÇÃO GRÁFICA E OPENGL 57
3.3.1 VISÃO GERAL 57
3.3.2 OPENGL 59
3.3.3 TRANSFORMAÇÕES GEOMÉTRICAS 61
3.3.4 QUATERNIÕES 64
3.3.5 CONCEITOS COMPLEMENTARES 65
3.3.6 MODELAGEM GEOMÉTRICA 66
3.4 APRENDIZADO POR REFORÇO 68

3.4.1 DEFINIÇÃO 69
3.4.2 ELEMENTOS DO APRENDIZADO POR REFORÇO 70
3.4.3 RETORNO AVALIATIVO 72
3.4.4 ALGORITMO Q-LEARNING 74
3.5 CONSIDERAÇÕES SOBRE O CAPÍTULO 77

4 O MODELO MATEMÁTICO DO DIRIGÍVEL 78
4.1 INTRODUÇÃO 78
4.2 PRINCÍPIOS BÁSICOS DE OPERAÇÃO DE DIRIGÍVEIS 79
4.2.1 FORÇAS AEROSTÁTICAS 79
4.2.2 FORÇAS AERODINÂMICAS E DINÂMICAS 81
4.2.3 FORÇAS DE PROPULSÃO 81
4.2.4 SUPERFÍCIES AERODINÂMICAS 82
4.3 OS TERMOS DO MODELO DINÂMICO 82
4.3.1 VETOR VELOCIDADE 83
4.3.2 MATRIZ DE MASSA 83
4.3.3 VETOR DE FORÇAS AERODINÂMICAS 83
4.3.4 VETOR DE GRAVIDADE E FLUTUAÇÃO 84
4.3.5 MODELO DA FORÇA DE PROPULSÃO 85
4.3.6 O EFEITO DO VENTO 85
4.3.7 TERMOS COMPLEMENTARES 86
4.4 CONSIDERAÇÕES SOBRE O CAPÍTULO 86

5 O SIMULADOR 87
5.1 A IMPLEMENTAÇÃO DO MODELO MATEMÁTICO 88
5.1.1 AS VARIÁVEIS DE ESTADO 89
5.1.2 A INTEGRAÇÃO DOS PASSOS DE SIMULAÇÃO 91
5.1.3 AS CLASSES IMPLEMENTADAS 93
5.1.4 A DETERMINAÇÃO DAS FORÇAS E DOS TORQUES 96
5.1.5 O TENSOR DE INÉRCIA 99
5.1.6 MÉTODOS DE INTEGRAÇÃO NUMÉRICA 100
5.1.7 OBSERVAÇÕES SOBRE A IMPLEMENTAÇÃO DO MODELO MATEMÁTICO .. 102
5.2 A RENDERIZAÇÃO DAS IMAGENS TRIDIMENSIONAIS 103
5.2.1 A RENDERIZAÇÃO DOS DIRIGÍVEIS 105

5.2.2 A RENDERIZAÇÃO DO TERRENO 107
5.2.3 A RENDERIZAÇÃO DE OBJETOS GENÉRICOS 110
5.3 A OFICINA DE CONSTRUÇÃO DE DIRIGÍVEIS 111
5.3.1 A CONFIGURAÇÃO DO ENVELOPE 112
5.3.2 A CONFIGURAÇÃO DA GÔNDOLA 113
5.3.3 A CONFIGURAÇÃO DOS PROPULSORES DE GÔNDOLA 115
5.3.4 A CONFIGURAÇÃO DOS LEMES DE CAUDA 116
5.3.5 A CONFIGURAÇÃO DO PROPULSOR DE CAUDA 117
5.3.6 A CONFIGURAÇÃO DOS COMPONENTES EMBARCADOS 119
5.4 A OPÇÃO DE EXTENSÃO DO SIMULADOR 120
5.4.1 O PROTOCOLO DE ACESSO REMOTO 122
5.5 TESTES E RESULTADOS OBTIDOS COM O SIMULADOR 124
5.5.1 AVALIAÇÃO DAS IMAGENS E DO DESEMPENHO GRÁFICO 125
5.5.2 AVALIAÇÃO DE DESEMPENHO DA UTILIZAÇÃO REMOTA 136
5.6 CONSIDERAÇÕES SOBRE O CAPÍTULO 137

6 DETERMINAÇÃO DE CAMINHOS 139
6.1 O AMBIENTE DE DETERMINAÇÃO DE CAMINHOS 139
6.2 DETERMINAÇÃO DE CAMINHOS USANDO PLANEJAMENTO 144
6.2.1 USANDO O ALGORITMO A* (A-ESTRELA) 144
6.3 DETERMINAÇÃO DE CAMINHOS USANDO APRENDIZADO POR REFORÇO 146
6.3.1 USANDO O ALGORITMO Q-LEARNING 147
6.4 CONSIDERAÇÕES SOBRE O CAPÍTULO 151

7 CONSIDERAÇÕES FINAIS 154
7.1 CONCLUSÕES 154
7.2 TRABALHOS FUTUROS 156

8 REFERÊNCIAS BIBLIOGRÁFICAS 158

9 APÊNDICE 165
9.1 APÊNDICE 1: MANUAL DE UTILIZAÇÃO DO SIMULADOR 166
9.1.1 A TELA PRINCIPAL 166
9.1.2 CRIANDO UM AMBIENTE FECHADO 167
9.1.3 CRIANDO UM AMBIENTE ABERTO 169

9.1.4 CRIANDO OS DIRIGÍVEIS 171
9.1.5 INICIANDO A SIMULAÇÃO 174

LISTA DE ILUSTRAÇÕES

FIG. 1.1 Alguns VANTs de destaque: (a) V-1; (b) Scout; (c) Pioneer; (d) Predator B. 18
FIG. 2.1 Esboço da arquitetura geral do projeto AURORA (RAMOS, 2002). 28
FIG. 2.2 Simulador do AS800 em Java/VRML (RAMOS *et al.*, 1999)..... 29
FIG. 2.3 Diagrama de Casos de Uso do Simulador do SDANT (PINHEIRO, 2006). 33
FIG. 2.4 Diag. Sequência do Caso de Uso "Monitorar Ambiente" (PINHEIRO, 2006)..... 34
FIG. 2.5 Diagrama de Classes do Agente (PINHEIRO, 2006). 34
FIG. 2.6 Fragmento do Diagrama de Metas (PINHEIRO, 2006). 35
FIG. 2.7 Interação entre os módulos do sistema de navegação (VIDAL, 2007). 37
FIG. 2.8 Fluxograma do processamento de imagens (VIDAL, 2007)..... 37
FIG. 2.9 Trajetória em situação com obstáculos no espaço 3D (VIDAL, 2007). 38
FIG. 3.1 Interação entre um agente e um ambiente. 44
FIG. 3.2 Robô Sojourner. 48
FIG. 4.1 Operação dos balonetes em um dirigível não rígido (RAMOS, 2002). 80
FIG. 4.2 Componentes principais para atuação em um dirigível (RAMOS, 2002)..... 82
FIG. 5.1 Algoritmo de execução do passo de simulação usando o método de Euler. 93
FIG. 5.2 Diagrama de classes do modelo matemático do dirigível. 94
FIG. 5.3 Um dirigível e seus componentes..... 97
FIG. 5.4 Comparação do método de Euler com a solução exata. 101
FIG. 5.5 As duas primeiras etapas para renderizar o envelope. 105
FIG. 5.6 Resultado final da renderização do envelope. 106
FIG. 5.7 Representação aramada e final de um dirigível com todos os componentes. 107
FIG. 5.8 Exemplos de mapa de relevo (a) e de textura (b) de um terreno. 108
FIG. 5.9 Terreno aramado (a) e texturizado (b) com fator de escala Z = 0. 109
FIG. 5.10 Terreno aramado (a) e texturizado (b) com fator de escala Z = 0,5. 109
FIG. 5.11 Terreno aramado (a) e texturizado (b) com fator de escala Z = 1. 110
FIG. 5.12 Ambiente fechado com objetos 3D estáticos e um dirigível. 111
FIG. 5.13 Janela de configuração do envelope de gás do dirigível. 112
FIG. 5.14 Alguns tipos de envelopes criados na oficina virtual. 113
FIG. 5.15 Janela de configuração da gôndola do dirigível. 114
FIG. 5.16 Diferentes configurações de gôndola feitas na oficina virtual. 114
FIG. 5.17 Janela de configuração dos propulsores de gôndola do dirigível. 115
FIG. 5.18 Diferentes configurações para os propulsores de gôndola. 116

FIG. 5.19 Janela de configuração dos lemes de cauda do dirigível. 117
FIG. 5.20 Diferentes configurações dos lemes de cauda do dirigível. 117
FIG. 5.21 Janela de configuração do propulsor de cauda do dirigível. 118
FIG. 5.22 Diferentes configurações do propulsor de cauda do dirigível. 118
FIG. 5.23 Janela de configuração dos equipamentos embarcados do dirigível. 119
FIG. 5.24 Janela de criação do dirigível. 120
FIG. 5.25 Ambiente urbano com 3 dirigíveis sobrevoando as ruas 125
FIG. 5.26 Texturas usadas no ambiente de simulação urbano da FIG. 5.25. 126
FIG. 5.27 Imagem do estádio obtida a partir da câmera embarcada no dirigível. 128
FIG. 5.28 Visão aérea do ambiente urbano criado a partir da TAB. 5.1. 129
FIG. 5.29 Visão aérea de um ambiente de floresta sobrevoado por 3 dirigíveis. 131
FIG. 5.30 Mapa de relevo (esquerda) e textura (direita) do ambiente de floresta. 131
FIG. 5.31 Região montanhosa desértica, com três dirigíveis pairando. 132
FIG. 5.32 Textura da região montanhosa. 133
FIG. 5.33 Ambiente de mar aberto, com três dirigíveis pairando. 135
FIG. 5.34 Mapa de relevo (esquerda) e textura (direita) do ambiente de mar aberto. 135
FIG. 6.1 Ambiente utilizado nos sistemas de determinação de caminhos 140
FIG. 6.2 Mapa de relevo utilizado no ambiente da FIG. 6.1. 141
FIG. 6.3 Decomposição tridimensional em células do ambiente da FIG. 6.1. 142
FIG. 6.4 Decomposição bidimensional em células do ambiente da FIG. 6.1 143
FIG. 6.5 Sistema de determinação de caminhos com o algoritmo A*. 145
FIG. 6.6 Sistema de determinação de caminhos com o algoritmo *Q-Learning*. 148
FIG. 6.7 Gráfico de convergência do algoritmo *Q-Learning*. 149
FIG. 6.8 Ambientes com caminhos determinados pelo *Q-Learning*. 152
FIG. 9.1 Tela principal do SimuBlimp. 166
FIG. 9.2 Janela de seleção de tipo de ambiente. 167
FIG. 9.3 Janela de criação de ambientes fechados (*indoors*). 168
FIG. 9.4 Janela de criação de ambientes abertos (*outdoors*). 169
FIG. 9.5 Botões da barra de ferramentas da janela principal 171
FIG. 9.6 Janela de solicitação do nome do dirigível a ser criado. 172
FIG. 9.7 Botões da barra de ferramentas da oficina virtual. 172
FIG. 9.8 Simulação em execução. 174

LISTA DE TABELAS

TAB. 5.1 Objetos inseridos no ambiente da FIG. 5.19. 126
TAB. 5.2 Avaliação do desempenho de renderização do ambiente urbano. 129
TAB. 5.3 Avaliação de desempenho com quantidade de objetos variável. 130
TAB. 5.4 Avaliação do desempenho de renderização do ambiente de floresta. 132
TAB. 5.5 Avaliação do desempenho de renderização do ambiente montanhoso. 133
TAB. 5.6 Avaliação de desempenho do ambiente de mar aberto. 134
TAB. 5.7 Avaliação de desempenho de uma consulta simples ao servidor de simulação. 136
TAB. 5.8 Avaliação da transmissão de imagens entre cliente x servidor. 137
TAB. 6.1 Execução do *Q-Learning* sobre o ambiente da FIG. 6.1. 148
TAB. 9.1 Comandos de teclado disponíveis durante a simulação. 175

LISTA DE SIGLAS

API	Application Programming Interface
AR	Aprendizado por Reforço
AS	Aprendizado Supervisionado
AURORA	*Autonomous Unmanned Remote mOnitoring Robotic Airship*
AUV	*Autonomous Underwater Vehicle*
BSD	*Berkeley Software Distribution*
CenPRA	Centro de Pesquisas Renato Archer
CFD	*Computational Fluid Dynamics*
CSG	*Construtive Solid Geometry*
DANT	Dirigível Autônomo Não Tripulado
DGPS	*Differential Global Positioning System*
IAD	Inteligência Artificial Distribuída
IAI	*Israel Aircraft Industries*
IME	Instituto Militar de Engenharia
INS	*Inertial Navigation Sensor*
IP	*Internet Protocol*
ISA	*International Standard Atmosphere*
GDL	Grau de Liberdade
GPS	*Global Positioning System*
HMD	*Head Mounted Display*
LEMS	Localização e Mapeamento Simultâneos
MaSE	*Multiagent Software Engineering*
MAV	*Micro Aerial Vehicle*
MIT	*Massachusets Institute of Technology*
NURBS	*Nonuniform Rational B-Splines*
OpenGL	*Open Graphics Library*
RDP	Resolução Distribuída de Problemas
RFC	*Request for Comments*
RL	*Reinforcement Learning*
SCA	Sistema de Coordenadas do Ambiente
SCL	Sistema de Coordenadas Local

SDANT	Sistema de Dirigíveis Aéreos Não-Tripulados
SLAM	*Simultaneous Localizing and Mapping*
SMA	Sistemas Multiagentes
TCP	*Transmission Control Protocol*
UAV	*Unmanned Aerial Vehicle*
UDP	*User Datagram Protocol*
ULV	*Unmanned Land Vehicle*
UML	*Unified Modeling Language*
VANT	Veículo Aéreo Autônomo Não Tripulado
VRML	*Virtual Reality Modeling Language*

RESUMO

Esta dissertação trata do desenvolvimento de um simulador para uma frota de dirigíveis robóticos autônomos não-tripulados, e é parte do projeto VANT (Veículo Aéreo Não-Tripulado) do Instituto Militar de Engenharia. O simulador desenvolvido é capaz de exibir cenários tridimensionais com ambientes e dirigíveis virtuais, sendo que o comportamento dos dirigíveis obedece a um modelo matemático que determina as reações do dirigível às forças dinâmicas, aerodinâmicas e deliberadas que atuam sobre o veículo. Um detalhamento de tal modelo é apresentado, bem como os passos para sua implementação. Outra característica do simulador é a interface de comunicação via soquetes, que possibilita a interação com sistemas externos, os quais podem consultar informações dos sensores embarcados simulados (como câmera, GPS, inclinômetro etc.) e comandar os atuadores (como os propulsores). Entre as aplicações potenciais para VANT estão: a vigilância, a inspeção e o reconhecimento de ambientes. Uma das vantagens oferecidas é o fato de o veículo atuar em altitudes elevadas, proporcionando cobertura privilegiada a alguns sensores, como a câmera. Dentre os tipos de VANT, o dirigível destaca-se como uma opção de baixo custo aquisitivo e operacional, possuindo, ainda, vantagens como: dispensar pista de pouso, capacidade de pairar no ar, navegar em baixa velocidade, entre outras. Ainda, por ser não-tripulado, é possível utilizar dirigíveis de menor porte, facilitando a logística operacional e reduzindo as conseqüências de possíveis incidentes. Sobremaneira, o uso desses veículos torna-se mais interessante quando possuem algum grau de autonomia, sendo capazes de realizar tarefas com o mínimo de intervenção humana. Entretanto, a conquista da autonomia requer o desenvolvimento de algoritmos inteligentes capazes de executar as tarefas inerentes ao problema, como navegação, pouso, decolagem, desvio de obstáculos, reconhecimento de objetos etc. Complementarmente ao simulador, são apresentados dois sistemas que o estendem com algoritmos inteligentes para determinação de caminhos entre dois pontos. Um deles baseia-se no algoritmo clássico A* e o outro, no algoritmo *Q-Learning*, sendo que o sistema baseado no *Q-Learning* é capaz de determinar trajetórias não só com distâncias mínimas, mas também com o mínimo de mudanças de direção. Os sistemas interagem com o simulador via soquetes, usando um protocolo pré-definido. A renderização das imagens 3D é feita utilizando a biblioteca OpenGL. São apresentados testes de desempenho que avaliam a utilização via soquetes e a renderização das imagens tridimensionais, e os resultados obtidos confirmam a viabilidade de se utilizar o simulador para o fim proposto.

ABSTRACT

This dissertation treats the development of a simulator for an unmanned autonomous robotic blimp team, and is part of the UAV (Unmanned Aerial Vehicle) project at the Military Institute of Engineering. The simulator developed is able to show 3D sceneries with virtual environments and blimps, since the behavior of the blimps obey a mathematic model that determines the reaction of the blimp to the dynamic, aerodynamic and deliberated forces that acts in the vehicle. A detailed specification of such model is presented, as well as the steps of its implementation. Another feature of the simulator is the communication interface through sockets, making possible the interaction with external systems, which can get information from simulated embedded sensors (like cameras, GPS, inclinometer etc.) and also control actuators (like propellers). Some potential applications for UAVs are: surveillance, inspecting and environment recognition. One of the advantages offered is that blimps can actuate in high altitudes, providing better coverage for some sensors, like the camera. Among the types of UAVs, the blimp appears as a low operational cost option, having, yet, advantages like: dispenses landing track, can float in the air, navigates at low speed etc. Still, as it is unmanned, it's possible to use smaller blimps, turning easier the operational logistic and reducing the consequences of possible incidents. Furthermore, the use of such vehicles becomes more interesting when they have some degree of autonomy, being capable to execute tasks with the minimum of human intervention. However, autonomy requires the development of intelligent algorithms to accomplish the intrinsic tasks of the problem, like navigation, landing, take-off, obstacle avoidance, object reckoning etc. Moreover, two another systems are presented, extending the simulator with intelligent algorithms that determine a path between two points. The first is based on the classic A* and the second, on the algorithm Q-Learning, so that the system based on Q-Learning is capable to determine trajectories with minimum path length and a minimum quantity of direction changes. Both systems interact with the simulator through sockets, using a pre-defined protocol. The 3D image rendering is accomplished using the OpenGL library. Performance tests are presented as well, evaluating the sockets communication and the 3D image rendering, and the final results confirm the viability in using the simulator for the proposal purpose.

1 INTRODUÇÃO

Os **V**eículos **A**éreos **N**ão **T**ripulados (VANTs, ou UAVs, do inglês ***U**nmanned **A**erial **V**ehicle*) vêm sendo pesquisados desde os primórdios da aviação. VANT é o termo usado para descrever todo e qualquer tipo de aeronave que não necessita de tripulação embarcada para ser guiada. Este tipo de aeronave pode ser controlado à distância, por meios eletrônicos e computacionais, sob a supervisão e governo humanos, ou sem a sua intervenção, utilizando computadores embarcados que executam de maneira autônoma as tarefas intrínsecas a uma missão. Assim como diversas outras tecnologias, a necessidade surgiu para a utilização militar. Atualmente, os VANTs são projetados e construídos para serem usados em missões perigosas, em que a execução por veículos aéreos tripulados coloque em risco a vida dos tripulantes. Outras aplicações incluem atividades de patrulhamento urbano, costeiro, ambiental e de fronteiras, atividades de busca e resgate, entre outras. O restante do capítulo apresenta diversas informações gerais sobre VANTs e posiciona o presente trabalho nesse contexto, incluindo os objetivos e resultados esperados pelo trabalho.

1.1 HISTÓRICO E VISÃO GERAL

Segundo a enciclopédia Wikipedia (WIKIPEDIA, 2007), as pesquisas sobre VANTs tiveram início antes mesmo do primeiro vôo de uma aeronave tripulada, que ocorreu em 13 de dezembro de 1903, pelos irmãos Wright. Há registros de que uma tecnologia primitiva em VANTs foi usada pelos Estados Unidos em duas guerras antes dessa data. Um desses registros data de fevereiro de 1863, dois anos antes do início da guerra civil americana, quando um inventor da cidade de Nova Iorque, chamado Charles Perley, registrou a patente de um bombardeiro aéreo não tripulado. O bombardeiro consistia em um balão de ar quente capaz de carregar um cesto de bombas atrelado a um detonador temporizado. Outro registro de uso de VANT data de 1883, quando o inglês Douglas Archibald utilizou uma pipa (ou "papagaio") para fazer fotos aéreas.

Durante a Primeira Guerra Mundial, os americanos perceberam o potencial militar dos VANTs e iniciaram uma pesquisa mais séria nessa área. Cerca de uma década após o fim da Primeira Guerra Mundial, o desenvolvimento de aviões sem tripulação vinha decaindo. Mas, em meados da década de 1930, novos VANTs surgiram como uma importante ferramenta de

treinamento em combate, e consistiam basicamente de veículos rádio-controlados utilizados como alvos. Na Segunda Guerra Mundial, os nazistas desenvolveram a V-1, uma bomba aérea lançada a partir de um ponto em terra, pré-programada para viajar por cerca de 240 quilômetros e cair como uma bomba normal.

Na década de 1960, durante a Guerra do Vietnã, os VANTs assumiram um papel bem mais importante do que o de veículo alvo para treinamento em combate. Os VANTs passaram a ser utilizados como veículo de reconhecimento e os primeiros passos começaram a ser dados para que se tornassem um veículo de ataque. Com a guerra fria no auge, a necessidade de imagens de reconhecimento de alta qualidade foi crescendo. Essa necessidade fez com que o governo americano encomendasse o desenvolvimento de uma aeronave rápida, furtiva e não tripulada. Disso surgiram o "AQM-34 *Ryan Firebee*" e o D-21. Na década de 1970, enquanto outras nações começaram a desenvolver seus próprios VANTs, os Estados Unidos passaram a focar outros tipos de VANTs. No fim da década de 1970, a força aérea israelense fazia um trabalho agressivo no desenvolvimento de VANTs. Em 1978, a *Israel Aircraft Industries* (IAI) construiu o *Scout*. O *Scout* consiste em um avião com cerca de 4 metros de envergadura, construído em fibra de vidro, com uma câmera embarcada com visão de 360º que pode transmitir imagens para uma estação em terra. Sua fuselagem em fibra de vidro lhe proporciona uma baixa assinatura em sistemas de radar, tornando-o também um veículo furtivo.

No final da década de 1980, Israel desenvolveu o *Pioneer*, um VANT leve, de baixo custo e de bom desempenho. Devido ao seu sucesso, as forças militares americanas compraram cerca de 20 *Pioneers*, que se tornaram os primeiros VANTs pequenos e de baixo custo da moderna força militar americana. A partir de então, as tecnologias em VANT passaram a se difundir e a se tornar mais comuns em outras nações, e outros veículos além do avião passaram a possuir versões não tripuladas.

Desde a década de 1990, os VANTs ocupam uma posição crítica e permanente no arsenal militar de alta tecnologia de várias nações. Além disso, também são usados para fins pacíficos, como monitoramento ambiental. Uma tendência que vem se confirmando é a diminuição de tamanho dos VANTs. Países como Estados Unidos, Grã-Bretanha, Coréia e Israel estão desenvolvendo os MAVs, sigla para *Micro Aerial Vehicle*. Devido ao seu

tamanho reduzido, o uso fica praticamente restrito a missões de monitoramento e vigilância. A FIG. 1.1 mostra alguns dos VANTs citados nos parágrafos anteriores.

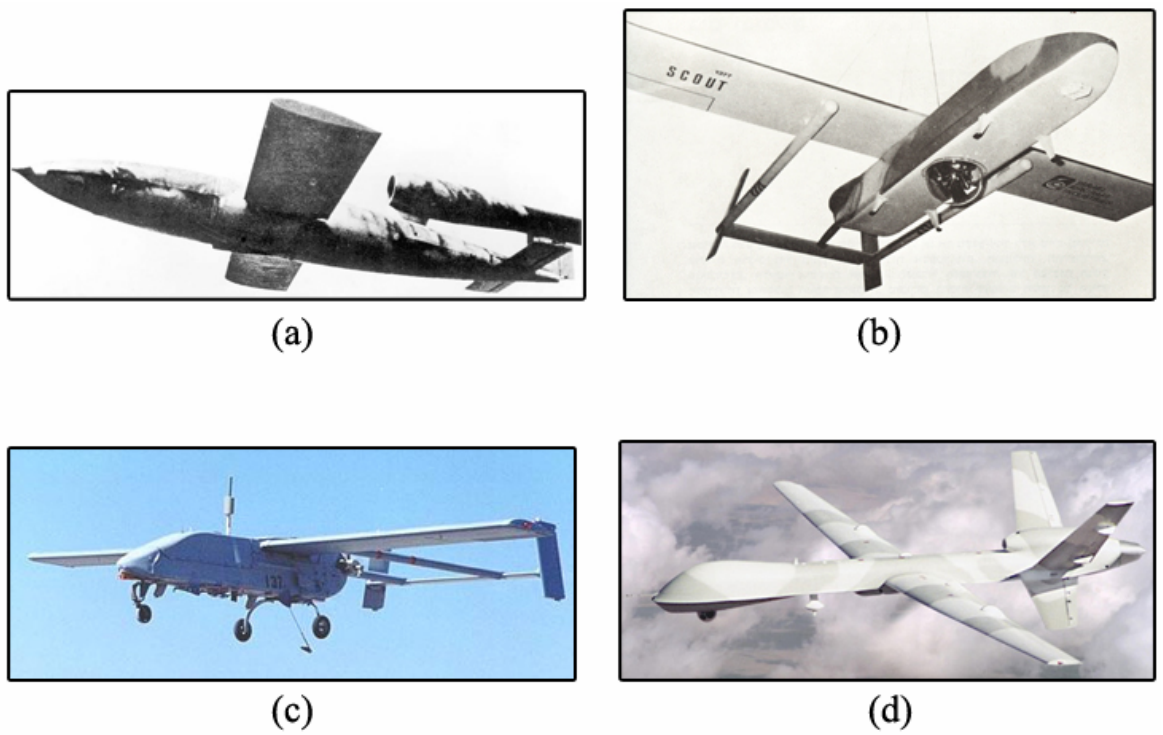

FIG. 1.1 – Alguns VANTs de destaque: (a) V-1; (b) Scout; (c) Pioneer; (d) Predator B.

No Brasil, um projeto precursor no desenvolvimento e pesquisa de VANTs é o projeto AURORA (*Autonomous Unmanned Remote mOnitoring Robotic Airship*), desenvolvido pelo CenPRA (Centro de Pesquisas Renato Archer). Resumidamente, o projeto AURORA visa o estabelecimento de tecnologias para a realização semi-autônoma de tarefas de monitoramento e inspeção ambiental, utilizando como veículo aéreo um dirigível. Por ter algumas características semelhantes ao problema geral abordado nesta dissertação, principalmente no que diz respeito ao tipo de veículo utilizado, o projeto AURORA é apresentado em mais detalhes no Capítulo 2. Outro projeto, da Universidade de São Paulo (USP), em São Carlos, apoiado pela Embrapa, objetiva o desenvolvimento de aeromodelos para a obtenção de imagens aéreas para monitoramento ambiental e agrícola.

Assim como em outros problemas de complexidade análoga, o problema de desenvolver um VANT pode ser mais bem atacado se decomposto em seus subproblemas inerentes, conforme (REIS, 2003). A complexidade do problema em questão se deve principalmente à necessidade de se proporcionar um determinado grau de autonomia ao VANT. Ainda, se os

veículos estiverem organizados em frotas, de forma que todos eles precisem agir para atingir um objetivo comum, a execução de tarefas autônomas pode demandar um alto nível de cooperação entre os veículos, o que também implica em uma complexidade adicional.

Com a necessidade de autonomia, as pesquisas se tornam ainda mais interdisciplinares e desafiadoras, por passarem a demandar o desenvolvimento de sistemas inteligentes para comandar, de forma remota ou embarcada, os veículos e seus componentes. Um exemplo de tarefa aparentemente simples que um VANT necessita realizar de forma autônoma é a navegação, que inclusive é uma tarefa comum a diversos outros problemas. No caso dos veículos aéreos, a navegação pode ser dividida em decolagem, pouso e deslocamento entre um ponto inicial e final (RAMOS *et. Al*, 2001). Com as tecnologias disponíveis atualmente, principalmente de sensoriamento remoto e de processamento de dados, o desenvolvimento de um sistema de navegação autônomo já deixou de ser algo notável, e sistemas com esse propósito são comuns na literatura subjacente. Contudo, o uso de determinados sensores aumenta demasiadamente o custo de construção do veículo e, na maioria das vezes, pode inviabilizar a finalização de um projeto ou a utilização de seu produto final.

A redução do custo de construção de uma frota de VANTs – a partir daqui chamada de sistema de VANTs – é fundamental para sua utilização em larga escala. Uma forma de reduzir o custo do sistema com um todo é reduzir e/ou distribuir o aparato tecnológico embarcado nos veículos, ou, utilizar essencialmente equipamentos de baixo custo. A escolha do veículo adequado também é de fundamental importância. Utilizando ferramentas e técnicas da Visão Computacional, é possível obter diversas informações do ambiente de atuação do veículo, usando apenas imagens desse ambiente. A obtenção dessas imagens pode ser feita a partir de câmeras embarcadas nos próprios veículos, e sendo a câmera um sensor de baixo custo quando comparada a outros tipos de sensores, o custo total do sistema pode ser reduzido. A escolha adequada do veículo também é essencial para a redução de custos, tanto de implementação como operacionais. O dirigível também facilita a realização bem sucedida de determinadas tarefas que podem ser complexas para serem executadas por outros veículos.

Quando o objetivo é a redução de custos, uma importante consideração é a realização de testes. Se um teste prático for mal sucedido, pode acarretar em sérias perdas materiais. Portanto, a necessidade de simular as missões antes de executá-las na prática é iminente, e isso requer um simulador capaz de representar as características físicas do ambiente e dos

veículos que atuam nesse ambiente. Um simulador também pode ser útil no desenvolvimento de algoritmos inteligentes para serem utilizados nos veículos, desde que seja capaz de fornecer as entradas necessárias para esses algoritmos e de aplicar corretamente as saídas dos algoritmos na simulação após o processamento dessas entradas.

O tema abordado pela dissertação tem aplicações relevantes e trata de tecnologias promissoras e emergentes, dando ao assunto um interesse adicional. A dissertação trata do desenvolvimento de um software simulador que fornece subsídios sensoriais para a elaboração de algoritmos inteligentes visando proporcionar crescentes graus de autonomia aos veículos. Através de um ambiente virtual tridimensional, que pode fazer uso de texturas e relevos de terrenos reais, o simulador é capaz de fornecer imagens de câmeras virtuais embarcadas em dirigíveis robóticos virtuais, bem como outros tipos de dados sensoriais, permitindo que outro sistema use esses dados para o desenvolvimento e testes de algoritmos inteligentes. Além das câmeras, são simulados outros sensores, como inclinômetros, acelerômetros e GPS (*Global Positioning System*). Também, é apresentado um sistema de determinação de caminhos para ambientes estruturados (com o algoritmo A*, da Inteligência Artificial) e desestruturados (com o algoritmo *Q-Learning*, de Aprendizado por Reforço) que usam como entradas dados oriundos do simulador.

1.2 MOTIVAÇÃO

Os veículos aéreos, principalmente quando autônomos e não tripulados, tem um enorme potencial de uso em diversas tarefas civis e militares. Dessas tarefas, podemos citar como principais as seguintes (RAMOS *et. al*, 2001) (RUSSEL e NORVIG, 2004):

- Reconhecimento de regiões hostis, onde aeronaves tripuladas colocariam em risco a vida dos tripulantes;

- Atividades de monitoramento em geral, como o monitoramento de tráfego urbano para auxiliar motoristas a seguirem por vias menos congestionadas;

- Vigilância e patrulhamento em geral, como a vigilância de áreas fronteiriças, costas, estádios de futebol, etc.;

- Pesquisa e monitoramento ambiental, como análises atmosféricas e climatológicas, detecção de queimadas em florestas, etc.;

- Aplicação de fluidos, como pulverização de lavouras e combate a incêndios;

- Transporte, tanto de pessoas quanto de objetos;

- Interligação e ampliação de redes de comunicação sem-fio, aumentando a cobertura do sinal e/ou interligando redes desconectadas; e

- Publicidade e propaganda, exibindo anúncios visuais e/ou sonoros em feiras e eventos, em lugares abertos, como estádios de futebol, ou fechados, como galpões.

Atualmente, grande parte das tarefas supracitadas é realizada de forma mais convencional, utilizando algum outro tipo de sensoriamento remoto, como sistemas de sensores fixos, satélites, aviões controlados por pilotos humanos, balões meteorológicos etc.. Entretanto, essas formas mais convencionais possuem algumas desvantagens. Os balões, por exemplo, não são manobráveis, não sendo possível definir com rigor a área a ser coberta durante seu vôo. Aviões controlados por pilotos humanos, embora permitam controlar a área a ser coberta, possuem tempo de vôo muito limitado, não só pelo combustível, mas também pelos recursos humanos utilizados, adicionando a preocupação com o conforto e a segurança da tripulação de bordo, e, conseqüentemente, aumentando o custo operacional. Além disso, requerem pistas de pouso e decolagem extensas para operarem. Os satélites, apesar de estarem bem avançados, enfrentam problemas com questões econômicas e políticas, além de sofrerem com problemas climatológicos. Os sensores fixos têm a desvantagem de cobrirem uma área limitada, sendo que a cobertura de grandes áreas requer a utilização de muitos sensores, aumentando muito o custo do sistema. Ademais, nem todos os ambientes permitem a instalação de um sensor fixo.

A utilização de uma frota de veículos aéreos não-tripulados autônomos pode solucionar, se não todos, grande parte dos problemas citados, desde que o sistema seja robusto e o tipo de veículo escolhido seja capaz de realizar determinadas tarefas de forma eficiente e com baixo custo. O custo de um sistema de VANTs pode variar muito, dependendo principalmente das tecnologias utilizadas em sua implementação. Além dos custos de implementação, os custos com testes também devem ser considerados, pois a maioria das operações são críticas e

precisam ser simuladas antes de serem postas em prática. Se o sistema real for utilizado para simulações, o custo será provavelmente igual ou superior ao custo da operação real.

Uma forma de contornar tais despesas seria usar simulação física em escala. Mas mesmo o uso de modelos em escala, possivelmente necessário, tem também um custo relativamente alto, pois, além dos modelos de veículo, tem de ser criado o modelo do ambiente, em escala compatível com a do modelo do veículo, com sensores, fontes de vento, obstáculos, correntes ascendentes etc., sob o completo domínio do pesquisador. Ainda, o volume do ambiente simulado é provavelmente muito grande e encarece ainda mais sua modelagem. A simulação física em escala corta custos e acelera o desenvolvimento, mas continua relativamente cara. Outra questão com testes reais é que um erro mínimo de cálculo em um sistema de navegação poderia resultar na queda de uma das aeronaves, causando grandes perdas materiais e, possivelmente, colocando em risco a segurança de pessoas. Testes reais também exigem uma logística de execução avançada, pois devem obedecer às leis que controlam o espaço aéreo e, por questões de segurança, devem ser realizados em áreas inabitadas.

O desenvolvimento de um simulador baseado em software é uma alternativa plausível, cujo custo se resume ao de um computador e do software necessário para seu funcionamento. A simulação pode reduzir o custo de desenvolvimento do sistema como um todo. É capaz também de reduzir o tempo para a realização de uma operação, pois uma simulação virtual tem a possibilidade de ser acelerada e melhor observada, sendo que os resultados podem ser tratados e apresentados em formato mais inteligível (FILHO, 1995).

Considerando-se que, na abordagem desta dissertação, o pouso, a decolagem, a detecção de obstáculos, a navegação entre dois pontos e o planejamento de trajetória são basicamente realizados através do processamento das imagens do ambiente captadas pelas câmeras embarcadas nos dirigíveis, todas essas tarefas podem ser realizadas valendo-se das imagens fornecidas pelo simulador. Apesar de, por questões de segurança, terem o uso desencorajado, outros sensores como o GPS (*Global Positioning System*), o INS (*Inertial Navigation Sensor*) e o SONAR (*Sound Navigation and Ranging*) também são simulados e podem ser consultados durante a simulação.

1.3 OBJETIVOS E CONTRIBUIÇÕES

Este trabalho é parte do projeto VANT do Instituto Militar de Engenharia, que tem por objetivo a construção de um sistema de veículos aéreos não-tripulados que possam atuar de forma autônoma em ambientes abertos (*outdoors*) como florestas, mar aberto ou áreas urbanas. Tal sistema deve ser capaz de realizar tarefas como vigilância, monitoramento e inspeção de ambientes, comunicação de dados, dentre outras.

O projeto VANT teve início com a dissertação de mestrado do Ten. Carlos Pinheiro, que propôs uma modelagem do sistema utilizando o paradigma de Sistemas Multiagentes (SMA) e a MaSE, uma linguagem específica para se trabalhar com SMA. Além disso, Pinheiro implementou um sistema de comunicação entre os agentes do sistema. Apesar de ser parte de um projeto já iniciado por (PINHEIRO, 2006), sendo, de certa forma, uma continuidade deste, o presente trabalho possui características que o diferenciam do trabalho desenvolvido anteriormente.

Um importante diferencial está no fato de os dirigíveis considerados serem heterogêneos, no que diz respeito às dimensões e ao aparato tecnológico dos mesmos. Essa heterogeneidade tem por objetivo principal uma redução ainda maior do custo do sistema proposto inicialmente por (PINHEIRO, 2006). Intuitivamente, a distribuição do aparato tecnológico viabiliza a utilização de veículos de menor porte e reduz o consumo de energia individual do dirigível, o que, conseqüentemente, reduz os custos de implementação e operação do sistema como um todo. Entretanto, o fato de os sensores estarem distribuídos em veículos diferentes indica que um dirigível pode precisar de uma informação proveniente de um sensor que ele não possui. Isso requer um mecanismo eficiente de comunicação e de compartilhamento de sensores e, logicamente, um maior grau de colaboração entre os veículos. Foge ao escopo da dissertação apresentar a definição formal detalhada de tal mecanismo, cuja modelagem inicial já foi abordada por (PINHEIRO, 2006).

Como objetivo específico, o trabalho é determinado à criação de um simulador com cenários tridimensionais, que forneça subsídios para o desenvolvimento de algoritmos inteligentes que, por sua vez, visam proporcionar autonomia aos veículos. Algumas características relevantes do simulador são:

- Renderização de fotos de terrenos reais na superfície do terreno simulado, como textura;

- Leitura de mapas de relevo que podem representar relevos de terrenos reais;

- Oficina de criação de dirigíveis, possibilitando a criação de dirigíveis com diversas configurações e parâmetros dinâmicos;

- Simulação dos efeitos dinâmicos do veículo (aplicação de forças e torques etc.) e do ambiente (vento, pressão atmosférica etc.);

- Possibilidade de importar objetos 3D complexos, como automóveis, construções, elementos naturais etc.;

- Possibilidade de simular os principais sensores utilizados para navegação, como GPS, INS, SONAR, inclinômetro etc.;

- Sistema de controle remoto, que permite controlar e obter informações dos dirigíveis e de seus sensores remotamente durante uma simulação, fazendo do simulador um servidor de simulação.

A implementação das características citadas requer o conhecimento de conceitos importantes de diversas áreas, como Robótica, Computação Gráfica, Física, Engenharia, Comunicação de Dados etc., e a união desses conceitos multidisciplinares em um único trabalho é uma das contribuições da presente dissertação. Seguindo o objetivo específico, o presente trabalho fornece, como principal contribuição, a descrição detalhada de todos os passos necessários para o desenvolvimento e uso de um simulador 3D, com dirigíveis robóticos virtuais que interagem entre si e com o ambiente em que atuam. Essa descrição abrange, além dos passos necessários para a obtenção das características visuais da simulação, os passos necessários para implementação das características comportamentais do dirigível, que utilizam um modelo matemático apropriado, adaptado do modelo desenvolvido por (GOMES e RAMOS, 1998).

De forma complementar, são apresentados dois sistemas que implementam algoritmos de determinação de caminhos entre dois pontos para ambientes estruturados e desestruturados. Tais sistemas também têm por objetivo mostrar a possibilidade de utilização do simulador

para o desenvolvimento de algoritmos específicos usando dados do simulador como entrada. A determinação de caminhos para ambientes estruturados (ou conhecidos) utiliza o algoritmo A*, clássico da Inteligência Artificial, por ser popular e ter bom desempenho. Para ambientes desestruturados, é usado o Aprendizado por Reforço. Especificamente, o algoritmo *Q-Learning* (WATKINS *et al.*, 1989) é utilizado para proporcionar o aprendizado de um caminho factível para ir de um ponto a outro no mapa do ambiente.

1.4 ESTRUTURA DA DISSERTAÇÃO

A dissertação está estruturada obedecendo ao padrão "introdução, desenvolvimento e conclusão". Neste sentido, o Capítulo 2 apresenta os trabalhos realizados dentro deste projeto e uma revisão bibliográfica sobre VANTs e robôs móveis, incluindo a utilização desses veículos na realização de tarefas específicas. Trabalhos relacionados à navegação baseada em visão também são apresentados. Alguns trabalhos relacionados à determinação de caminhos usando o Aprendizado por Reforço também são comentados.

Visando preparar o leitor à multidisciplinaridade inerente deste trabalho, no Capítulo 3, é apresentado, em síntese, o arcabouço teórico necessário ao desenvolvimento de VANTs robóticos autônomos. São abordados os temas Agentes, Robótica, Computação Gráfica e Aprendizado por Reforço.

No Capítulo 4, é apresentado o modelo matemático de um dirigível e as adaptações realizadas para utilizá-lo no simulador desenvolvido. Os termos do modelo matemático são apresentados em detalhes individualmente.

O Capítulo 5 apresenta o simulador desenvolvido. São apresentados diversos detalhes sobre sua implementação. São mostrados detalhes de implementação da dinâmica, da representação gráfica dos objetos simulados, da renderização do terreno, dentre outras coisas. Também são apresentados alguns testes de desempenho com o simulador.

Como complemento, o Capítulo 6 apresenta um sistema de determinação de caminhos que interage com o simulador, fornecendo uma possível trajetória entre dois pontos quaisquer do mapa de relevo utilizado pelo simulador. São apresentadas soluções para ambientes estruturados e para ambientes desestruturados, bem como os resultados obtidos com tais soluções.

No Capítulo 7, são feitas as considerações finais. São apresentadas as conclusões que puderam ser tiradas no decorrer do trabalho, bem como algumas sugestões para trabalhos futuros.

No Capítulo 8, são apresentadas as referências bibliográficas usadas no decorrer do desenvolvimento da parte escrita e prática desta dissertação. Fechando, no Capítulo 9, subseção 9.1, é apresentado o Anexo 1, que consiste em um manual de utilização do simulador, mostrando todas as opções de configuração, formas de operação etc.

2 REVISÃO DE LITERATURA

Neste capítulo são apresentados alguns dos importantes trabalhos relacionados ao desenvolvimento de VANTs encontrados na literatura, bem como aqueles que dizem respeito ao desenvolvimento de soluções para os subproblemas inerentes ao problema geral. Trabalhos que tratam de simulação, robôs móveis autônomos e navegação baseada em visão computacional também são apresentados.

2.1 TRABALHOS RELACIONADOS

O projeto que deu início à pesquisa em VANT com dirigíveis no Brasil e gerou artigos ainda hoje referenciados na literatura nacional e internacional é o projeto AURORA – *Autonomous Unmanned Remote mOnitoring Robotic Airship* – iniciado pelo CenPRA em 1996. O projeto AURORA tem como objetivo o estabelecimento de tecnologia para a operação semi-autônoma de um dirigível não-tripulado, almejando seu uso como plataforma aérea para aplicações em inspeção, pesquisa e monitoração ambiental, climatológica e de biodiversidade. No projeto AURORA, visa-se o estabelecimento de dirigíveis não-tripulados com significativos graus de autonomia durante todas as fases de suas missões, incluindo a habilidade de planejar e executar sensoriamento e navegação, diagnosticar e recuperar-se de falhas, e adaptativamente replanejar missões baseando-se na avaliação, em tempo real, de informação sensorial e de restrições ambientais. O projeto AURORA gerou diversos trabalhos científicos, dos quais os principais são comentados mais adiante.

Em (ELFES *et al.*, 1998) foram descritos alguns aspectos importantes do projeto. É descrita a plataforma física do projeto, que consiste em um dirigível não-tripulado acompanhado remotamente por uma estação móvel terrestre. O equipamento embarcado no dirigível conta com um PC104, sensores como inclinômetro, câmera e um receptor GPS. Há também uma especificação da arquitetura de software e da interface homem-máquina. O trabalho também dá uma visão superficial da modelagem dinâmica e do sistema de controle, da percepção e navegação baseada em visão e de navegação autônoma. Também são apresentados alguns argumentos a favor da utilização de dirigíveis em vez de aviões ou helicópteros em tarefas como monitoramento e inspeção ambiental. Dentre os argumentos citados estão o baixo custo de operação, a capacidade de pairar no ar, a baixa turbulência, a

capacidade de pousar e decolar na vertical e o baixo consumo de combustível. A FIG. 2.1 mostra um esboço geral do funcionamento do AURORA. Como se pode ver na figura, o dirigível é assistido por uma estação em terra, responsável também por tratar os dados coletados pelo dirigível.

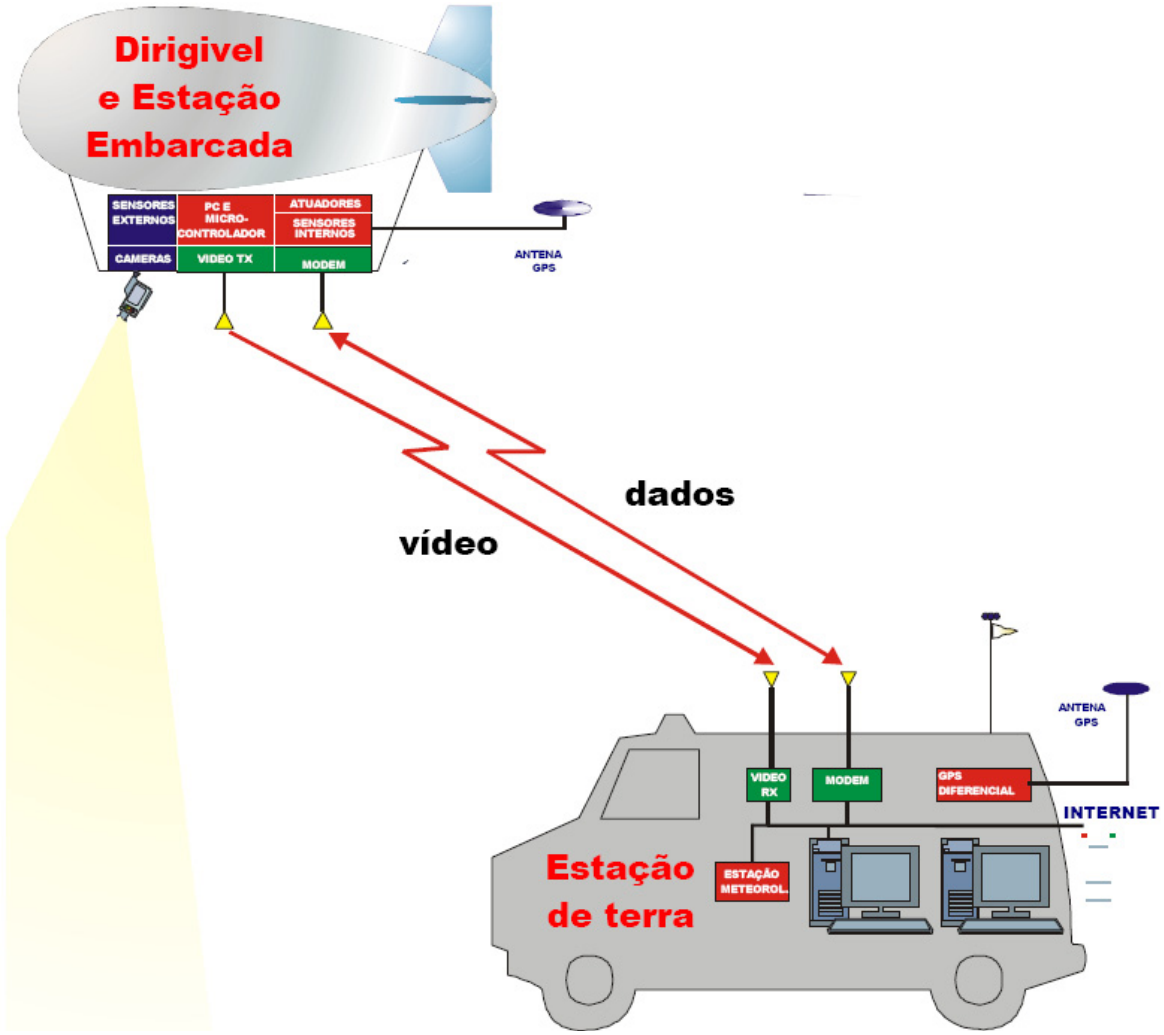

FIG. 2.1 – Esboço da arquitetura geral do projeto AURORA (RAMOS, 2002).

O trabalho de (GOMES e RAMOS, 1998) forneceu um ponto de partida para pesquisas na área de robótica utilizando dirigíveis aéreos, pois descreveu de forma mais específica os princípios físicos e o modelo matemático do dirigível modelo YEZ-2A, que utiliza um balonete interno para auxiliar no controle de altitude do vôo. O autor também descreve o sistema de vôo do dirigível e as conseqüências de efeitos externos, como a temperatura.

Alguns softwares desenvolvidos para dar suporte ao projeto AURORA são apresentados em (RAMOS *et al.*, 1999). Os autores descrevem um conjunto de ferramentas para projeto e teste de métodos de controle e navegação, visualização do dirigível, compartilhamento de informações via internet e treinamento de pilotos, desenvolvidos com a ferramenta MATLAB/Simulink. Um simulador 3D em Java/VRML também é apresentado, juntamente com um estudo de caso de um vôo simulado controlado manualmente de forma remota. A FIG. 2.2 mostra o simulador feito em Java/VRML.

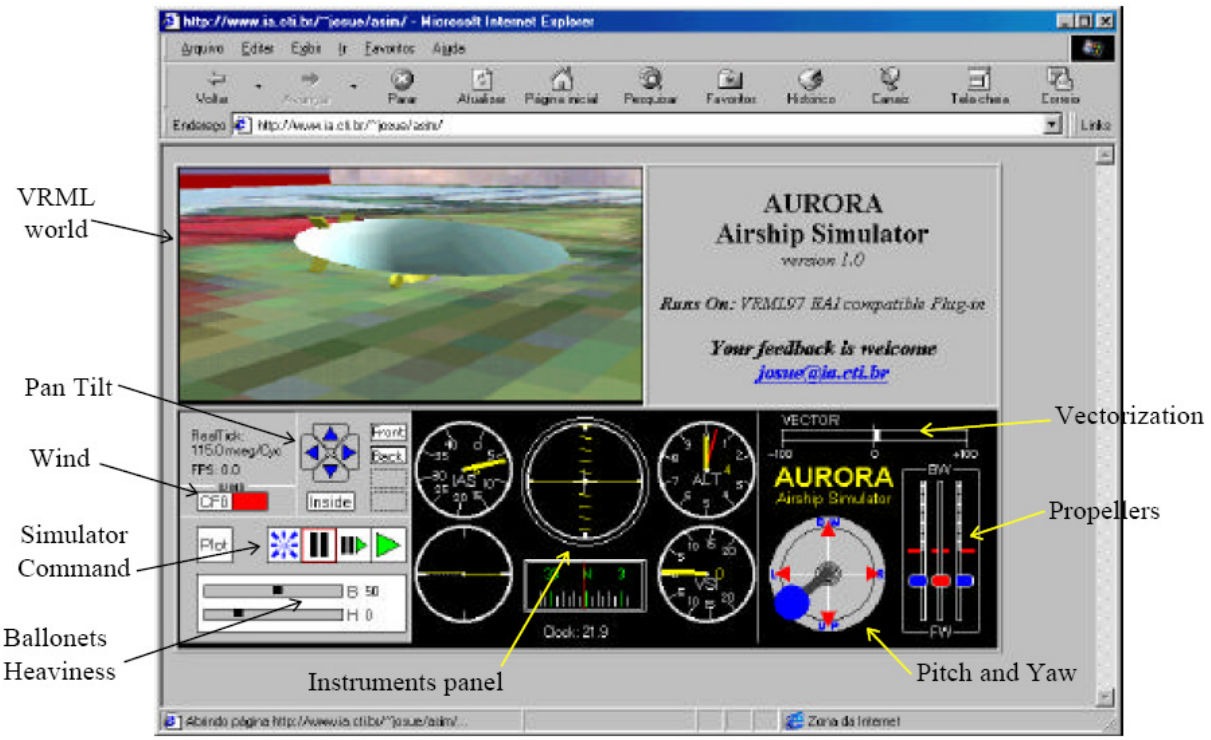

FIG. 2.2 – Simulador do AS800 em Java/VRML (RAMOS *et al.*, 1999).

Em (RAMOS *et al.*, 2001), é descrito como um dirigível não-tripulado percorre uma trajetória pré-definida usando dados oriundos de sensores como GPS, acelerômetro, sensor de vento e inclinômetro, sem a utilização de técnicas de Visão Computacional. São apresentados os resultados de testes do seguimento de uma trajetória guiado por um sistema embarcado, com a altitude sendo controlada manualmente de forma remota. O modelo matemático do dirigível AS800 também é descrito, bem como toda a plataforma experimental utilizada.

Uma grande contribuição ao projeto AURORA, resultante da união dos esforços anteriores, é o trabalho de (RAMOS, 2002), que consiste em uma tese de doutorado abordando o desenvolvimento e a implementação de um protótipo de dirigível robótico não-

tripulado, com capacidade de realizar vôos autônomos reais, seguindo trajetórias definidas por pontos de passagem. Para contemplar esse objetivo, foram realizadas modificações estruturais na mecânica e aerodinâmica do dirigível utilizado, adequando-o ao uso como veículo robótico. No trabalho, também foram estabelecidas as infra-estruturas, embarcada e em terra, de componentes de software e de hardware, envolvendo sensores, atuadores, processadores e software de tempo real. Ainda, um modelo dinâmico aprimorado do dirigível foi apresentado, considerando notadamente os aspectos de força de propulsão e influência da dinâmica do vento e, com base neste novo modelo, foram desenvolvidos ambientes de simulação em MATLAB/Simulink e Java/VRML, sendo este último baseado no simulador apresentado anteriormente em (RAMOS *et al.*, 1999). Foi implementado também um ambiente de suporte ao desenvolvimento e à operação do dirigível robótico, bem como um sistema de controle e navegação, compreendendo algoritmos de seguimento de trajetória e de perfil de altitude entre pontos de passagem. Tal sistema foi analisado e seus parâmetros de sintonia foram ajustados tanto em simulação quanto em vôos reais. Finalmente, foi estabelecido um protótipo de arquitetura de software robótico para o dirigível, testado apenas em simulação, contemplando aspectos deliberativos e reativos de uma missão robótica completa.

Ainda no contexto do projeto AURORA, o problema da identificação do modelo dinâmico longitudinal do dirigível foi atacado de três formas diferentes por (FARIA, 2005) em sua dissertação de mestrado. Foram feitas a identificação estacionária, a identificação dinâmica e linearizada do veículo a partir de um vôo com entradas de perturbações conhecidas e, por fim, a identificação por meio de estratégias evolutivas. Essas abordagens foram desenvolvidas utilizando o simulador do dirigível AS800 apresentado por (RAMOS *et al.*, 1999).

A quantidade e a qualidade dos trabalhos desenvolvidos no contexto do projeto AURORA fizeram dele um marco no desenvolvimento de VANTs no Brasil, tanto que é um projeto altamente referenciado na literatura acadêmica e um dos poucos em que há uma implementação real bem sucedida e avançada do sistema. Entretanto, um ponto do projeto que pode ser aperfeiçoado é o fato de o mesmo fazer uso de apenas um veículo. Isso faz com que o veículo deva ser de grande porte, para suportar todo o aparato tecnológico necessário à realização de uma missão. Conseqüentemente, o veículo tem autonomia baixa devido ao consumo excessivo de bateria pelo grande número de componentes embarcados. Além disso, a execução de uma missão incorre em maior custo e maior risco de insucesso, pois é atribuída

a uma única aeronave e, caso ela sofra um acidente ou uma pane, a missão terá de ser interrompida.

No que diz respeito à simulação de veículos interagindo com ambientes, um trabalho muito referenciado na literatura, inclusive por trabalhos do projeto AURORA, é o de (BRUTZMAN, 1995), que trata do desenvolvimento de um simulador tridimensional para um submarino autônomo de exploração científica. O simulador implementa o modelo matemático do submarino e do ambiente (hidrodinâmica), além de um protocolo de rede que permite o uso remoto do simulador com a finalidade de criar um ambiente de colaboração entre pesquisadores.

Outro trabalho que pode ser citado é o trabalho de (SOUSA *et al.*, 2004), que é uma versão condensada de (VARAYA, 2004). Ambos dizem respeito a um sistema que simula o ataque de uma equipe de VANTs heterogêneos (quanto ao tamanho, armamento, velocidade etc.) a alvos de diferentes tipos (baterias antiaéreas, radares, depósitos de armas etc.) localizados no solo. O problema é estruturado em dois níveis: o de planejamento e o de execução. O nível de planejamento consiste em analisar o cenário e estabelecer rotas e cronogramas de ataques, de forma que uma medida de risco seja minimizada. A execução se baseia numa hierarquia de controladores de 4 níveis, do controlador de tarefa – responsável pela missão como um todo – ao controlador de VANT – responsável pela liberação de armas e por manobras elementares. A arquitetura do sistema é interessante, entretanto o simulador serve apenas para visualizar o processo de ataque, e não como fonte de dados para a realização de algum processamento necessário à execução do ataque.

O trabalho de (SINOPOLI *et al.*, 2001) expõe um sistema de navegação de VANT baseado em visão. O VANT é equipado com uma câmera embarcada, possuindo estimativas de seu estado através de sensores GPS/INS. A missão do VANT é a navegação em baixa altitude, em um ambiente 3D parcialmente conhecido, desviando-se de obstáculos e tentando minimizar a distância até um objetivo. Ainda no âmbito da Visão Computacional, há o trabalho de (SAEEDI *et al.*, 2006), que descreve um sistema baseado em visão para a localização tridimensional de um robô móvel em um ambiente desconhecido. O sistema inclui 3 câmeras embarcadas e sua principal característica é a estimação do movimento do robô sem o conhecimento de cenas anteriores, marcações na superfície ou sensores extras. O trabalho de

SAEEDI apresenta resultados interessantes, principalmente com relação ao desempenho do sistema, mas não informa detalhes de como obteve tal desempenho.

Em (ZAMSTEIN *et al*., 2006), o autor apresenta o processo de aprendizado de um robô autônomo deliberativo. Trata-se de um pequeno robô móvel, de nome Koolio, que possui duas câmeras e dois sonares, como sensores, e carrega um pequeno refrigerador com comidas e bebidas. O robô-garçom utiliza aprendizado por reforço para determinar os melhores caminhos até os locais onde ele é chamado. O método *Q-Learning*, apresentado na subseção 3.4.4 da presente dissertação, é utilizado no processo de aprendizado do robô Koolio.

2.2 TRABALHOS ANTERIORES

Como citado em subseções anteriores, esta dissertação articula-se com outros esforços para a realização de um projeto mais amplo, cujo objetivo geral está descrito na subseção 1.3. Essa reunião de esforços tem como antecedente a dissertação de mestrado desenvolvida por (PINHEIRO, 2006), que, *strictu sensu*, trata da modelagem de um simulador de uma frota de veículos aéreos autônomos não-tripulados. Mais especificamente, (PINHEIRO, 2006) realizou a modelagem de um Sistema de Dirigíveis Aéreos Não-Tripulados, denominado SDANT.

O problema é abordado utilizando o paradigma de Sistemas Multiagentes. Foram definidos os requisitos funcionais e não-funcionais e desenvolvidos diagramas de caso de uso, de seqüência, de papéis, de tarefas concorrentes, de classes e de implantação. Dentre os requisitos identificados, podem-se citar como principais: evitar colisões, corrigir a trajetória e realizar o deslocamento da frota automaticamente. Enquanto a frota de dirigíveis pode, com mais especificidade, ser caracterizada como um sistema multirobô, o simulador é um sistema multiagente, constituído de agentes de software, que apresenta a troca de mensagens entre os diferentes componentes do sistema.

O trabalho de (PINHEIRO, 2006) trata então da modelagem do simulador segundo o paradigma multiagente, visando à construção de um sistema multiagente composto de agentes de software, para resolver o problema geral de construção de uma frota de veículos aéreos autônomos não-tripulados cooperativos para comunicação de dados e monitoramento de ambientes desestruturados. Além da modelagem em si, o trabalho trata da avaliação do

modelo produzido, mostrando o desenvolvimento de subsistemas do simulador a partir do modelo. A FIG. 2.3 mostra um modelo de casos de uso do simulador desenvolvido.

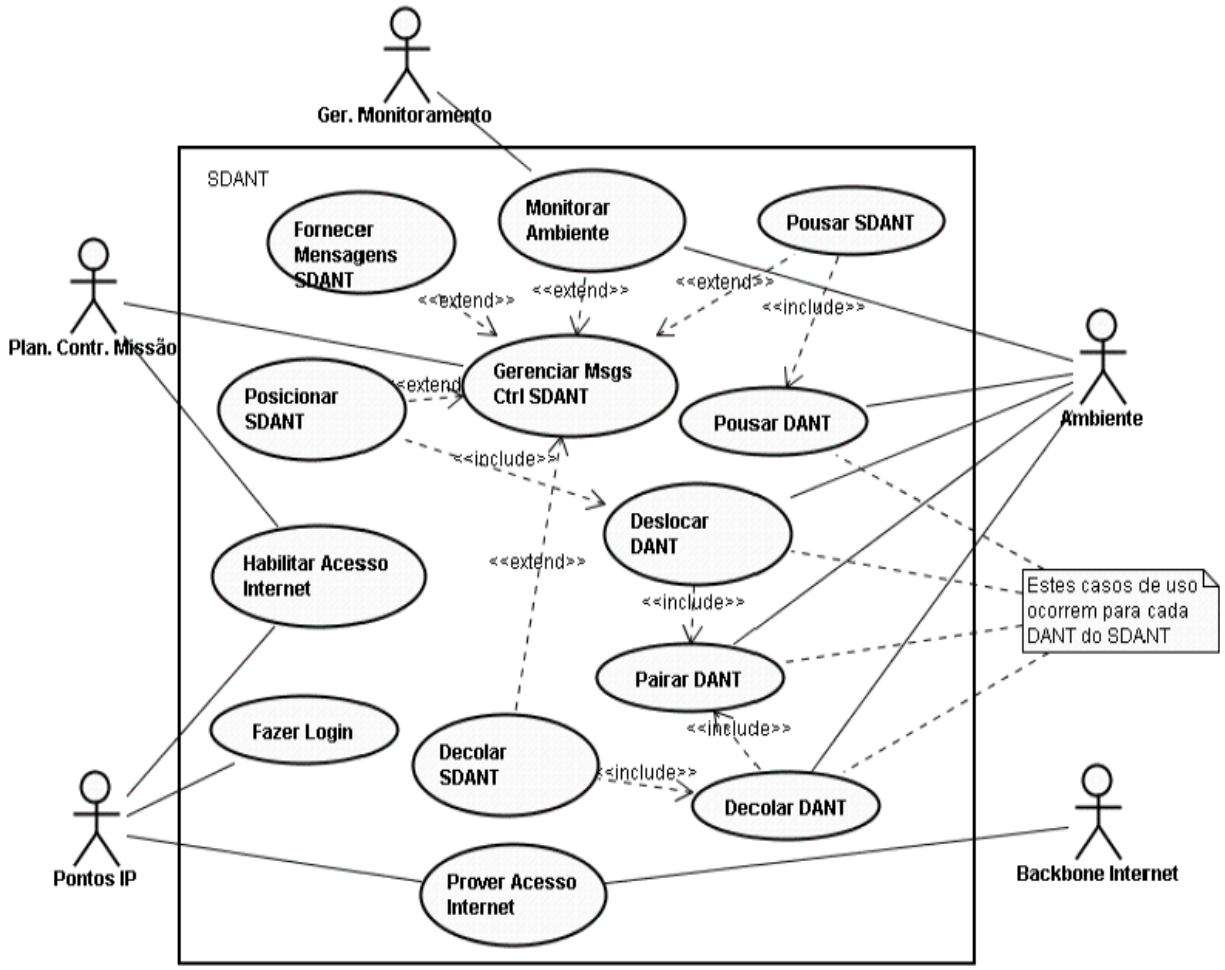

FIG. 2.3 – Diagrama de Casos de Uso do Simulador do SDANT (PINHEIRO, 2006).

Como é mostrado na FIG. 2.3, os casos de uso modelados por (PINHEIRO, 2006) foram: “Monitorar Ambiente”, “Prover Acesso à Internet”, “Fornecer Mensagens do SDANT”, “Decolar SDANT”, “Pousar SDANT”, “Posicionar SDANT”, “Gerenciar Mensagens de Controle do SDANT”, “Fazer Login”, “Habilitar Acesso Internet”. Todos os casos de uso possuem seu respectivo diagrama de seqüência. A FIG. 2.4 apresenta o diagrama de seqüências do caso de uso “Monitorar Ambiente”.

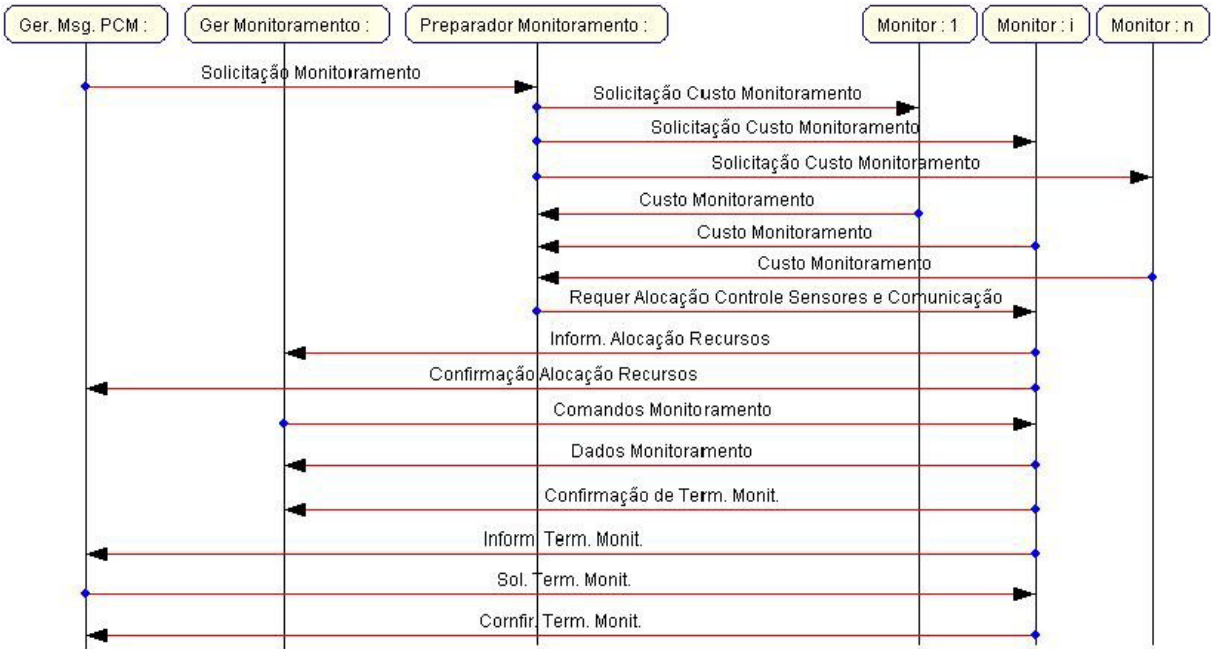

FIG. 2.4 – Diag. Sequência do Caso de Uso "Monitorar Ambiente" (PINHEIRO, 2006).

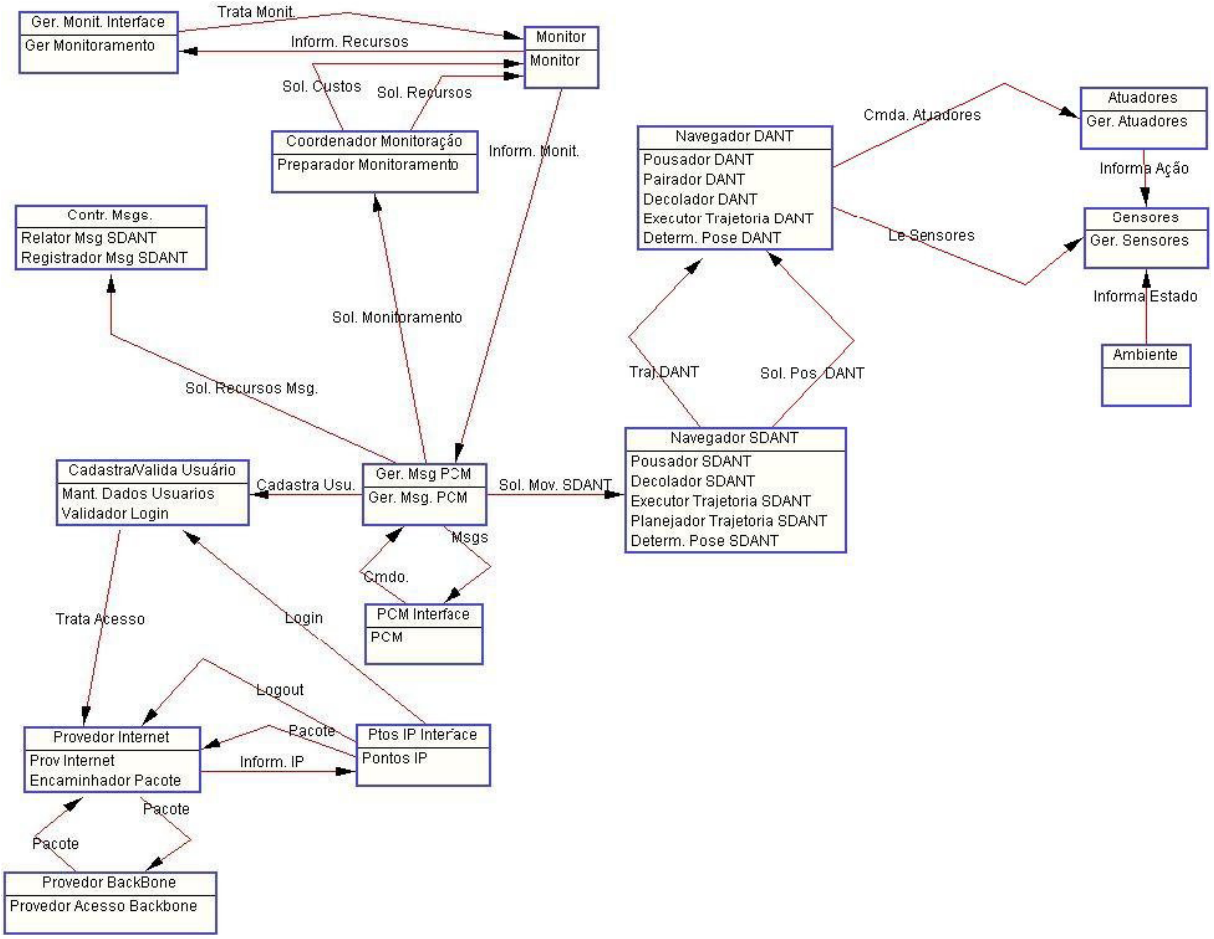

FIG. 2.5 – Diagrama de Classes do Agente (PINHEIRO, 2006).

A modelagem proposta por Pinheiro é realmente bem completa, e contém ainda: o diagrama de papéis de todo o sistema, cerca de quarenta diagramas de tarefas concorrentes e quarenta diagramas de classes de conversação, o diagrama de classes do agente, entre outros, sendo que todos obedecem aos padrões da MaSE. O diagrama de classes do agente é mostrado na FIG. 2.5. Além dos diagramas estáticos, um mecanismo de troca de mensagens entre os agentes do sistema foi implementado. Para isso, Pinheiro utilizou a JADE, que é uma estrutura de software para a construção, implementação e operação – regidas pelas especificações FIPA relativas a sistemas multiagentes, inteligentes e inter-operáveis – de aplicações baseadas em agente.

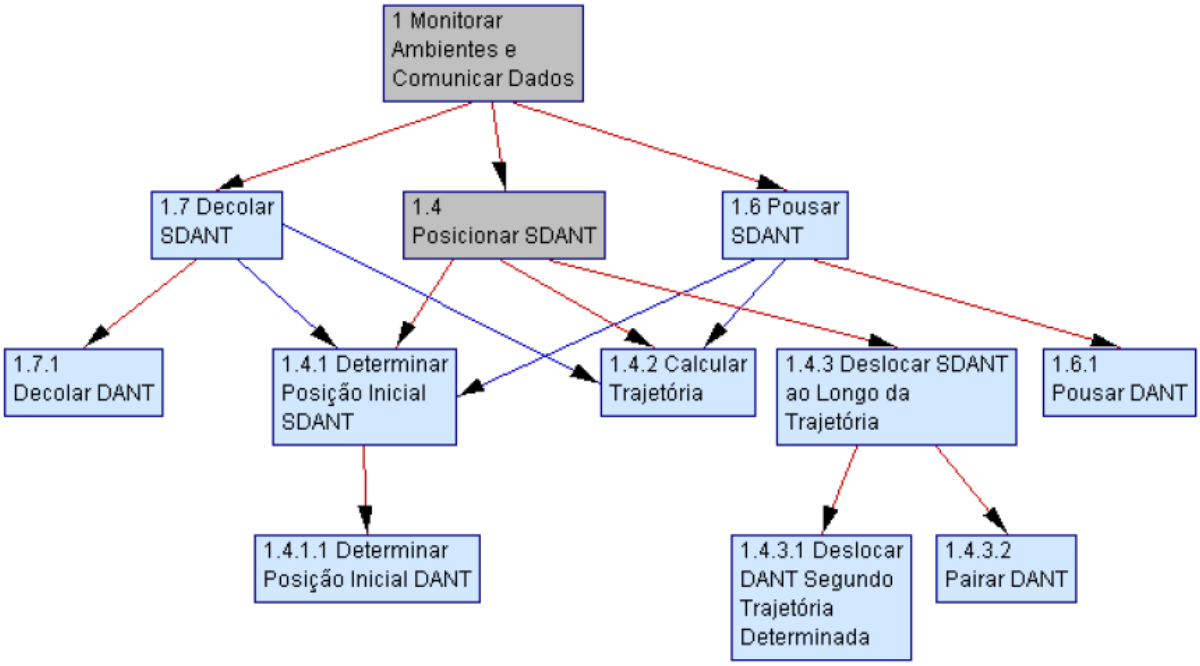
1 Monitorar Ambientes e Comunicar Dados
1.7 Decolar SDANT
1.4 Posicionar SDANT
1.6 Pousar SDANT
1.7.1 Decolar DANT
1.4.1 Determinar Posição Inicial SDANT
1.4.2 Calcular Trajetória
1.4.3 Deslocar SDANT ao Longo da Trajetória
1.6.1 Pousar DANT
1.4.1.1 Determinar Posição Inicial DANT
1.4.3.1 Deslocar DANT Segundo Trajetória Determinada
1.4.3.2 Pairar DANT

FIG. 2.6 – Fragmento do Diagrama de Metas (PINHEIRO, 2006).

O trabalho de (VIDAL, 2007) também é parte dos esforços para a solução do problema geral abordado na presente dissertação. Em seu trabalho, Vidal propõe um sistema de navegação para dirigíveis não-tripulados baseado em visão. Vidal guiou-se pelo sistema de navegação intrínseco no fragmento do diagrama de metas da FIG. 2.6, atendo-se principalmente nos três casos de uso relacionados, que são:

- Posicionar SDANT: comanda o posicionamento dos DANT. Este posicionamento segue os seguintes passos: determinar a posição inicial do SDANT; calcular a trajetória;

deslocar o SDANT ao longo da trajetória determinada; e pairar SDANT. Pré-Condição: SDANT deve pairar;

- Decolar SDANT: comanda a decolagem dos DANTs. Esta decolagem segue os seguintes passos: decolagem do DANT Âncora (o DANT que abstrai o SDANT); em seguida, o DANT que lhe é adjacente; e assim sucessivamente. Após decolarem, os DANTs pairam a 100 metros de altura. O SDANT define também, ao comandar a decolagem de cada DANT, a posição que cada DANT deve ocupar imediatamente após a decolagem. Os DANTs não podem mover-se para uma eventual posição de destino sem que respeitem a restrição da intercomunicação deles. Pré-Condição: SDANT deve estar ativo;

- Pousar SDANT: comanda o pouso dos DANTs. O pouso segue os seguintes passos: determinar a posição inicial do SDANT; calcular a trajetória; deslocar o SDANT ao longo da trajetória determinada; e pousar SDANT. Pré-Condição: SDANT deve pairar.

Um detalhamento do funcionamento do sistema de navegação empregado no projeto VANT do IME, iniciado por Pinheiro, foi realizado por Vidal, pois tal detalhamento se mostrou fundamental para a viabilização do atendimento de alguns requisitos não-funcionais do SDANT, como evitar colisões, corrigir a trajetória e realizar o deslocamento da frota automaticamente. Assim, após análise do funcionamento do sistema em questão, ele foi dividido nos seguintes módulos:

- Módulo de visão: responsável por processar as imagens oriundas dos sensores;

- Módulo de comunicação: responsável por enviar e receber mensagens para os outros membros da equipe;

- Módulo de localização: responsável por determinar a localização do dirigível e dos objetos detectados pelo módulo de visão;

- Módulo de planejamento de trajetória: responsável por definir a trajetória de um dirigível, a partir de dados gerados pelo módulo de localização;

- Módulo de controle: responsável por determinar as ações dos atuadores do dirigível para que esse se movimente seguindo determinada trajetória.

A FIG. 2.7 dá uma visão geral da interação entre os módulos do sistema de navegação proposto por Vidal.

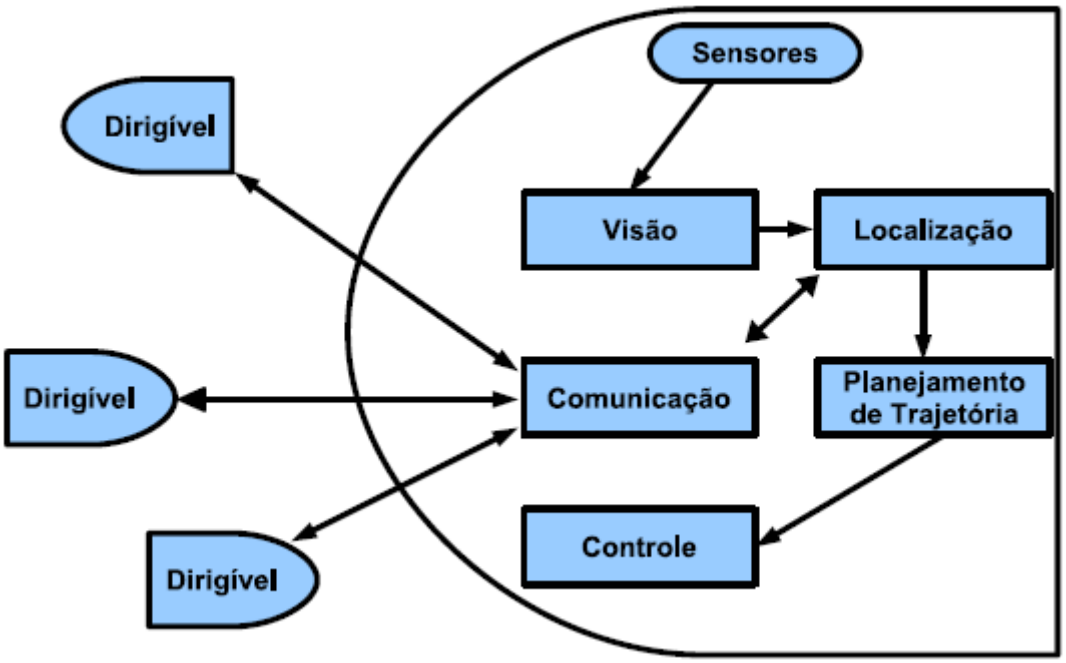

FIG. 2.7 – Interação entre os módulos do sistema de navegação (VIDAL, 2007).

O sistema de navegação proposto consiste primeiramente em capturar as imagens extraídas de câmeras embarcadas nos dirigíveis. Posteriormente, as imagens são analisadas com o uso de técnicas de visão estereoscópica para se criar uma representação interna do ambiente. A FIG. 2.8 apresenta um diagrama de atividade em alto nível dessas primeiras etapas.

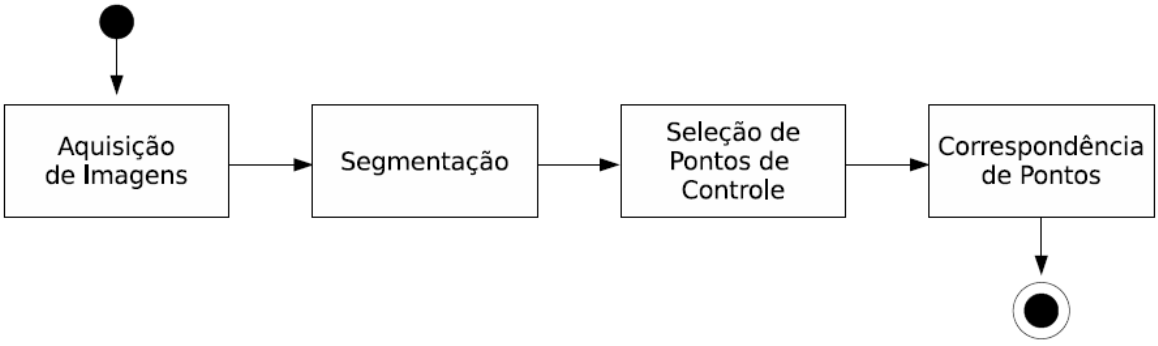

FIG. 2.8 – Fluxograma do processamento de imagens (VIDAL, 2007).

O trabalho de Vidal também abordou diversos algoritmos para processamento de imagens, necessários à reconstrução tridimensional do ambiente a partir de um par de imagens estéreo. Nos testes realizados, Vidal constatou que a biblioteca OpenCV teve melhor desempenho que outras metodologias testadas e, na plataforma Intel, obteve melhores resultados, o que o levou a escolher tal plataforma para ser utilizada na implementação dos algoritmos relacionados com a parte de visão computacional do sistema de navegação. Esses resultados compreendem o desenvolvimento de algoritmos aplicáveis nas primeiras fases da reconstrução tridimensional do ambiente.

Após obter uma representação, mesmo que parcial, do ambiente, é possível realizar o planejamento de trajetória para determinar um caminho ao ponto conhecido mais próximo do objetivo. Nesse sentido, (VIDAL, 2007) apresenta também um sistema de planejamento de trajetórias para ambientes de duas e três dimensões que faz uso da meta-heurística "Colônia de Formigas". O ambiente, total ou parcialmente reconstruído, é decomposto em células e passa a ter uma representação discretizada, de forma a reduzir o tamanho do espaço de busca.

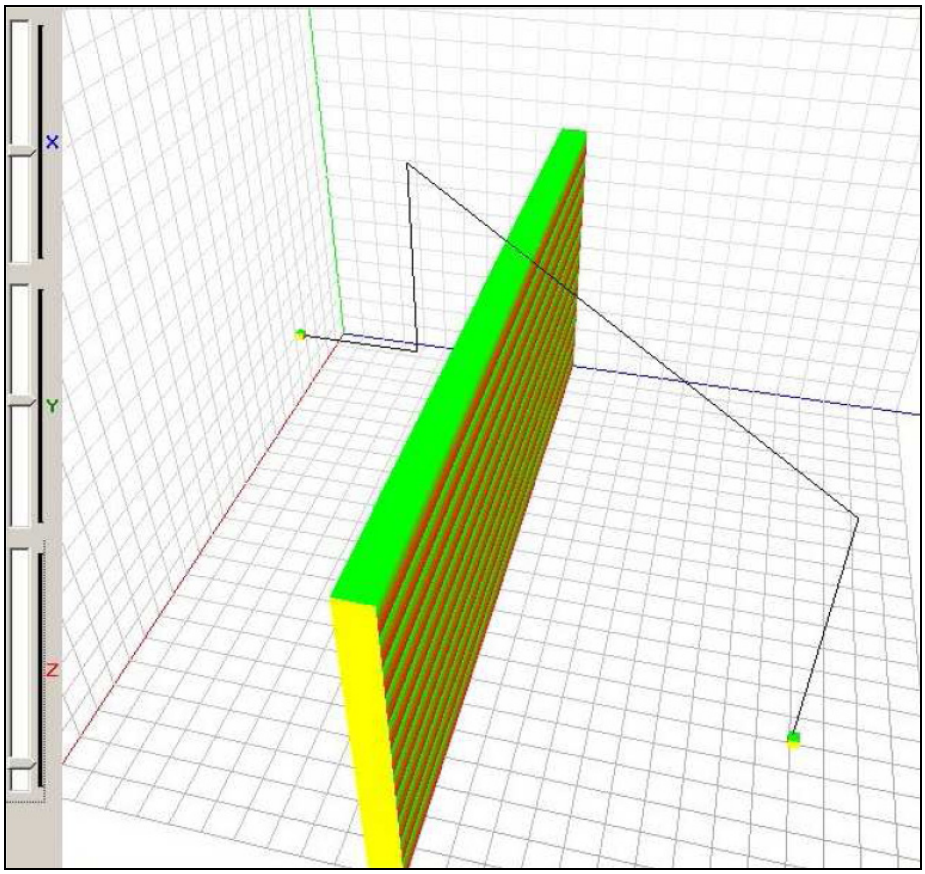

FIG. 2.9 – Trajetória em situação com obstáculos no espaço 3D (VIDAL, 2007).

A trajetória planejada no trabalho de VIDAL consiste em um caminho até uma célula objetivo, composto por uma seqüência de células adjacentes. Tal seqüência considera apenas células desocupadas, ou seja, que não correspondem a obstáculos. O sistema ainda é capaz de refazer o caminho em tempo real caso um obstáculo móvel sobreponha o caminho previamente planejado. A FIG. 2.9 mostra um exemplo de uma trajetória planejada a partir da decomposição em células de um ambiente tridimensional.

A continuidade do projeto para a solução do problema geral apresentado nesta dissertação é uma questão importante a ser considerada. Como continuidade imediata, há o trabalho do 1º Tenente Mendonça, do Instituto Militar de Engenharia, que consiste no desenvolvimento de algoritmos inteligentes para executar a navegação autônoma dos dirigíveis e que, até a data de conclusão desta dissertação, se encontrava em andamento. Por ser um projeto de interesse do Instituto Militar de Engenharia, a aquisição de dirigíveis reais de pequeno porte é iminente. Com a aquisição de dirigíveis reais, seria possível realizar testes reais e ajustar o simulador do presente trabalho de forma a tornar a simulação mais real, aumentando a fidelidade do comportamento dos dirigíveis virtuais com relação aos dirigíveis reais.

Há muitos outros trabalhos relacionados com o desenvolvimento de veículos aéreos autônomos não-tripulados, principalmente no que tange à navegação e ao controle do veículo, abordando ambientes previamente conhecidos ou não. Qualquer tentativa de descrever, mesmo que sucintamente, cada um ou uma pequena parte desses trabalhos, ocuparia diversas páginas. A descrição dos trabalhos supracitados se deve ao fato de eles terem elementos de grande semelhança com os do presente trabalho.

Antes de entrar especificamente no assunto do presente trabalho, se faz necessário o conhecimento de alguns conceitos gerais de outras áreas. O capítulo seguinte apresenta um embasamento teórico para o entendimento do restante da dissertação. São abordados quatro assuntos principais, que são: Agentes, Ambientes e Sistemas Multiagentes; Robótica; Computação Gráfica e OpenGL; e, Aprendizado por Reforço.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é dedicado a fazer uma descrição dos principais conceitos utilizados no desenvolvimento desta dissertação, com o intuito de proporcionar ao leitor um melhor entendimento da problemática e da solução aqui tratadas. São abordados quatro assuntos: Agentes, Ambientes e Sistemas Multiagentes; Robótica; Computação Gráfica e OpenGL; e, Aprendizado por Reforço. Vale ressaltar que as subseções a seguir não objetivam esgotar os respectivos assuntos. O objetivo aqui é elucidar os principais conceitos que se encontrem dentro do contexto do presente trabalho.

3.1 AGENTES, AMBIENTES E SISTEMAS MULTIAGENTES

Agentes são personagens computacionais que atuam em um ambiente de acordo com um *script* definido, direta ou indiretamente, por um usuário. Eles podem atuar isoladamente ou em comunidade, formando sistemas multiagentes. Em relação a esses últimos, questões como comunicação e negociação devem ser consideradas.

3.1.1 AGENTES

Atualmente, não existe uma definição para "agente" aceita por toda a comunidade da Inteligência Artificial. Uma definição genérica foi proposta por (FERBER *et. al*, 1991):

> "Um agente é uma entidade real ou virtual, capaz de agir num ambiente, de se comunicar com outros agentes, que é movida por um conjunto de inclinações (sejam objetivos individuais a atingir ou uma função de satisfação a otimizar); que possui recursos próprios; que é capaz de perceber seu ambiente (de modo limitado); que dispõe (eventualmente) de uma representação parcial deste ambiente; que possui competência e oferece serviços; que pode eventualmente se reproduzir e cujo comportamento tende a atingir seus objetivos utilizando as competências e os recursos que dispõe e levando em conta os resultados de suas funções de percepção e comunicação, bem como suas representações internas."

A definição acima é bem genérica, podendo compreender tanto um robô móvel (entidade real) como um robô virtual (entidade computacional). O agente se caracteriza por perceber seu ambiente através de sensores e de agir nesse ambiente através de atuadores. O termo percepção diz respeito às entradas perceptivas do agente em qualquer momento dado. A seqüência de percepções do agente é a história completa de tudo que o agente percebeu. Uma ação do agente pode ser baseada apenas na percepção atual que ele teve, ou em toda a

seqüência de percepções passadas. Em sua forma mais simples, um agente obtém a percepção de um determinado conjunto de sensores e toma uma decisão baseada apenas nessa percepção, ignorando qualquer tipo de acontecimento que não seja a percepção atual. Chamamos esse tipo de agente de “Agentes Reativos Simples”. Os Agentes Reativos Simples têm a admirável propriedade de serem simples, mas se caracterizam também por terem uma inteligência muito limitada, e são funcionais somente quando a decisão correta puder ser baseada apenas na percepção atual do ambiente.

Quando se tem um agente que deve atingir um determinado objetivo, pode-se complementar a percepção atual quando essa não for suficiente para que o agente tome a decisão correta em uma determinada situação. Uma boa forma de complementar essa simples percepção seria utilizar um modelo do ambiente e realizar um planejamento baseado nesse modelo, bem como no histórico de percepções. Adicionar-se-ia uma nova característica ao agente, que é o fato de ele passar a planejar o futuro e de considerar sua experiência. Basicamente, o agente simula diversas situações, criando um tipo de estrutura decisória, geralmente baseada em simulações.

Uma estrutura decisória armazena informações sobre conseqüências de ações tomadas em determinadas situações. Assim, o agente poderá realizar a ação que lhe proporcione um futuro melhor, não mais considerando apenas o resultado imediato dessa ação. Entretanto, na maior parte dos casos, não se pode ter um modelo completo e perfeito do ambiente, o que impede de realizar um planejamento que garanta que o agente sempre tomará a melhor decisão em uma determinada situação. Uma forma de tentar superar essa dificuldade é implementar algum tipo de mecanismo que dê ao agente a capacidade de aprender com o seu passado, ou seja, com as ações anteriormente tomadas. Pode-se considerar como exemplo um agente que tem o objetivo de fazer previsão do tempo. Se ele se basear apenas na percepção atual das condições do tempo, ele não será capaz de prever o que está por vir.

Como não é possível criar um modelo perfeito do ambiente em que o agente se encontra, uma boa maneira de melhorar a capacidade de previsão do agente é desenvolver uma função de aprendizado. Considere-se que os sensores do agente consigam obter diversas informações sobre as condições climáticas atuais, como a velocidade do vento, a temperatura, a umidade do ar, a densidade das nuvens etc., e que tenha existido uma situação climática semelhante no passado. O agente poderia se basear nessa situação para determinar a previsão atual. Isso

remete ao fato de que o agente deva ser capaz de manter um conhecimento das experiências anteriores, criando uma estrutura de diversas combinações de variáveis (p.ex., condições climáticas) que levam a um determinado estado (p.ex., previsão do tempo). Essa seria uma forma de fazer com que o agente aprendesse a partir dos dados, através de mecanismos de associação e recuperação de memórias.

O método anterior é uma das formas de aprendizado mais conhecida. Entretanto, existe outra forma de aprendizado, chamada "Aprendizado por Reforço", que se mostra mais eficaz na solução deste tipo de problema. No Capítulo 6, o Aprendizado por Reforço é utilizado para atacar o problema de determinação do caminho mínimo entre dois pontos. Esses tipos de agentes, que têm sua decisão baseada não apenas na percepção atual, mas também em ações passadas e no modelo do ambiente, são chamados de "agentes cognitivos".

3.1.2 AMBIENTES

Um elemento de extrema importância quando se trabalha com agentes é o ambiente. Em linhas gerais, um ambiente é uma instância de um problema solucionável por um agente racional. Existem diversos tipos de ambientes, e cada um deles afeta diretamente o projeto do programa que determina o comportamento do agente. Ao projetar um ambiente, a primeira etapa deve ser sempre especificar o ambiente de tarefa da forma mais completa possível. Existe uma diversidade de ambientes de tarefa que podem surgir. É possível identificar alguns aspectos semelhantes desses ambientes, de forma que se possa classificá-los e dividi-los em categorias distintas. Algumas definições informais são (RUSSEL e NORVIG, 2004):

- *Completamente Observável* x *Parcialmente Observável*: se os sensores detectam todos os aspectos relevantes para a escolha da ação, de acordo com uma função que mede o desempenho do agente, ele é completamente observável. Um ambiente poderia ser parcialmente observável se os sensores não fossem capazes de perceber todas as informações relevantes sobre o ambiente.

- *Determinístico* x *Estocástico*: se o próximo estado do ambiente é completamente determinado pelo estado atual e pela ação executada pelo agente, dizemos que o ambiente é determinístico. Caso contrário, ele é estocástico.

- *Episódico* x *Seqüencial*: em ambientes episódicos, as ações tomadas em um dado momento não afetam os estados posteriores, e também não são baseadas nos estados

anteriores. Um exemplo seria um agente que classificasse parafusos como perfeitos ou defeituosos. Um ambiente seqüencial é aquele onde as ações são interdependentes, ou seja, a ação atual é baseada no histórico de ações anteriores e afeta as ações futuras, como em um jogo de Xadrez, por exemplo.

- *Estático* x *Dinâmico*: se um ambiente puder se alterar independente de o agente estar ou não realizando ações, dizemos que ele é dinâmico. Caso contrário, ele é estático. Um ambiente de um táxi autônomo é completamente dinâmico e o de um jogo de xadrez é estático.

- *Discreto* x *Contínuo*: um ambiente discreto possui um número finito de estados. As percepções e as ações também podem ser discretas, ou de número finito. O xadrez é um exemplo de ambiente discreto, com um conjunto discreto de ações e percepções. O problema do táxi autônomo possui um ambiente contínuo, ou seja, com infinitos estados. O conjunto de ações e percepções também é infinito.

- *Agente único* x *Multiagente*: a distinção entre esses tipos de ambientes é bastante simples. Quando o problema é resolvido por um único agente dizemos que ele é de agente único. Caso contrário, ele é multiagente. Em um ambiente multiagente, os agentes podem ser competitivos, como em um jogo de xadrez, onde cada jogador procura reduzir o desempenho do oponente, ou cooperativos, como em um canteiro de obras, onde a cooperação é fundamental para que não aconteçam problemas para atingir o objetivo.

Analisando as classificações acima, percebe-se que o caso mais difícil é quando o ambiente é: *parcialmente observável*, *estocástico*, *seqüencial*, *dinâmico*, *contínuo* e *multiagente*.

A FIG. 3.1 mostra a interação de um agente com um ambiente. Nessa figura, as características relevantes do ambiente são percebidas pelos sensores do agente e, este último, por sua vez, processa essas informações e realiza a ação buscando atingir seu objetivo, conseqüentemente, alterando o estado do ambiente.

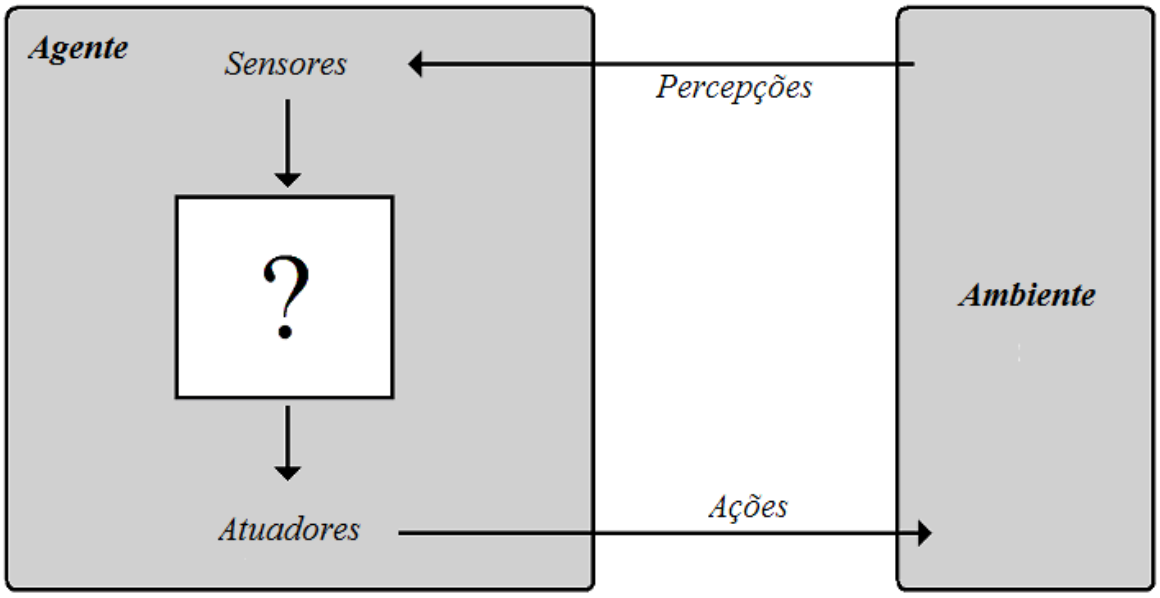

FIG. 3.1 – Interação entre um agente e um ambiente.

3.1.3 SISTEMA MULTIAGENTES

O assunto *Sistemas Multiagentes* (SMA) é parte da Inteligência Artificial Distribuída (IAD), que por sua vez se completa com o tópico "Resolução Distribuída de Problemas" (RDP). Resumidamente, o que diferencia SMA de RDP é que, na RDP, é considerada a existência de um problema preciso que deve ser resolvido, e não existem agentes desenvolvidos previamente. Os problemas abordados em SMA são as atividades de um conjunto de agentes autônomos em um universo multiagentes.

Um agente que atua isoladamente em um ambiente tem, em princípio, um controle total sobre os resultados de suas ações. Pode também utilizar mecanismos de planejamento (no caso de um agente cognitivo) para construir a melhor seqüência de ações para atingir seus objetivos. Caso o agente, por outro lado, esteja imerso num ambiente onde coexistam outros agentes, certamente irá ocorrer uma interferência social entre esses agentes. Conseqüentemente, processos mais complexos de coordenação e negociação podem ser utilizados. Esses aspectos, bem como o restante da teoria sobre SMA, são descritos resumidamente a seguir, conforme (REZENDE, 2005):

- *Interferência Social*: a coexistência de dois agentes com objetivos próprios, em um mesmo ambiente, pode resultar em uma interferência social positiva quando uma ação qualquer de um agente aproximar o outro agente de seu objetivo. Caso a ação afaste o

outro agente de seu objetivo, a interferência social é negativa. Caso não haja influência, a interferência é neutra. Pode ser interessante analisar os efeitos mútuos das ações dos dois agentes. Esse fenômeno é chamado de *interação social*. A interferência social existe mesmo que os agentes envolvidos não estejam conscientes dos objetivos uns dos outros. Obviamente, quando estes fatos forem representados na mente dos agentes, eles podem raciocinar socialmente para tirar proveito da existência dos outros, como descrito a seguir;

- *Autonomia, Delegação, Adoção, Compromisso e Cooperação*: uma conseqüência fundamental da interferência social é a alteração dos objetivos atingíveis por um determinado agente. Se um agente só puder atingir determinado objetivo na presença de outro agente, diz-se que o primeiro não é *autônomo* para atingir um determinado objetivo, sendo *dependente* do segundo. Para esse caso, a Teoria da Dependência diz que um agente deve raciocinar de modo a delegar a ação que necessita ao segundo agente, garantindo que este último realize a ação, consciente de que está ajudando o primeiro a alcançar um objetivo. Nesse sentido, o primeiro pode tentar realizar um planejamento social para *influenciar* o segundo a *adotar* o objetivo de realizar uma determinada ação para que o primeiro atinja seu objetivo. Uma vez aceita a adoção, o segundo agente firma um *compromisso* junto ao primeiro para realizar a ação desejada. Uma possibilidade é tentar fazer uma troca de ações, do tipo "uma mão lava a outra", ocasionando uma *cooperação* – quando o objetivo de ambos é o mesmo – ou um *escambo social* – quando os objetivos são diferentes;

- *Negociação e Protocolos*: os agentes envolvidos em um processo de influência e adoção podem eventualmente divergir em relação a certos aspectos, como requisitos de qualidade, tempo de início e de término das ações, ordem de ativação, eventuais custos e benefícios associados etc. Torna-se necessário, portanto, um processo de *negociação*, cuja definição é dada por: "Negociação é um processo de tomada de decisão conjunta. É comunicação, direta ou implícita, entre indivíduos que estão tentando chegar a um acordo para benefício mútuo. O significado original da palavra é apenas fazer negócios, mas negociação também é a atividade central na diplomacia, na política, na religião, no direito e na família. A negociação engloba conversações sobre controle de armas, a interpretação de textos religiosos e disputa de guarda de crianças. Todos negociam." A automação dos processos de negociação requer o estabelecimento de um *protocolo* de interação, que

preestabeleça etapas definidas na troca de mensagens entre os agentes, garantindo a convergência do processo;

- *Coordenação*: em situações de interferência social, escambo social e cooperação, os agentes devem necessariamente se coordenar para agir e para decidir a ordem de execução das ações. Decidir coisas como: qual agente realizará qual ação, como os agentes irão trocar as informações sobre o resultado da execução das mesmas, como terão acesso a recursos escassos, como irão eventualmente alterar a propriedade de suas ações em função das ações dos outros etc. Essa coordenação talvez seja uma das tarefas mais complexas. Similarmente ao processo de negociação, devem ser utilizados protocolos específicos para esse fim. Para diminuir a complexidade desse processo, padrões de interação que se repetem muitas vezes são capturados em estruturas preestabelecidas, definindo o que se chama de *organizações*;

*S*ão descritos sucintamente a seguir os tipos (ou padrões) de interação (REZENDE, 2005):

- *Neutralismo*: não ocorre interação entre os agentes;

- *Competição*: ambos os agentes são atingidos negativamente, pois competem pelos mesmos recursos;

- *Amensalismo*: um agente é afetado pela ação não proposital de outro agente;

- *Parasitismo*: um agente (parasita) depende de outro (hospedeiro) para sua existência, afetando-o negativamente, porém sem causar a morte imediata do prejudicado;

- *Predação*: uma das metas do agente (predador) é a eliminação do outro (presa);

- *Comensalismo*: a interação beneficia apenas um dos agentes (comensal), mas não prejudica o outro (hóspede);

- *Proto-cooperação*: a interação otimiza a obtenção de suas metas, embora não seja vital para os agentes;

- *Simbiose*: a interação entre os agentes é obrigatória para a obtenção de suas metas.

Uma organização de um SMA pode ser vista de forma simplificada como um conjunto de restrições adotadas por um grupo de agentes para que possam atingir seus objetivos globais mais facilmente. Existem ainda linguagens e ambientes padronizados para lidar com SMA. Entretanto, foge ao escopo desta dissertação esgotar o assunto SMA, que por si só é tema de pesquisa que gera teses e dissertações. Maiores detalhes sobre SMA podem ser obtidos em (REZENDE, 2005).

Como já mencionado, os agentes podem ser reais, ou seja, podem ter uma constituição física. Muitas aplicações que usam robôs são abordadas através de SMA, em que o robô é um agente físico. Para fins de modelagem e implementação de sistemas inteligentes, os VANTs podem ser considerados agentes robóticos, como já idealizado por (PINHEIRO, 2006). A subseção a seguir aborda o tema Robótica, fornecendo os principais conceitos relacionados a esta dissertação.

3.2 ROBÓTICA

Definindo de forma simples, um robô consiste em "um agente físico que executa tarefas manipulando o mundo físico" (RUSSEL e NORVIG, 2004). Para essa definição valer, os robôs devem ser capazes de exercer forças físicas sobre o ambiente. Isso é feito através dos efetuadores (ou atuadores), como pernas, rodas, articulações, garras, propulsores etc. Para um robô atuar em seu ambiente, ele também deve ser capaz de percebê-lo, tarefa essa conseguida através de sensores, como câmeras e sensores ultra-som, além de giroscópios e acelerômetros para aferirem o movimento do próprio robô.

Os robôs atuais podem ser enquadrados em uma dentre três principais categorias: manipulador, robô móvel ou híbrido, sendo esta última uma combinação das duas categorias anteriores. Os manipuladores, ou braços robóticos, estão fisicamente fixados em um ponto de seu local de trabalho. O movimento de um manipulador envolve a movimentação de uma cadeia de articulações controláveis, permitindo posicionar seus efetuadores em diferentes pontos dentro do local de trabalho.

Os robôs móveis, a segunda categoria de robôs, podem se deslocar por seu ambiente usando pernas, rodas ou mecanismos semelhantes. Esses robôs podem ser controlados remotamente, podem ser autônomos ou parcialmente autônomos. Dentre os tipos de robôs

móveis mais importantes estão: os veículos terrestres não-tripulados ou ULV (*Unmanned Land Vehicle*), os veículos aéreos não-tripulados (citados no início da dissertação), os veículos autônomos subaquáticos ou AUV (*Autonomous Underwater Vehicle*) e os viajantes planetários, como o famoso robô *Sojourner*, que explorou a superfície de marte em julho de 1997, mostrado na FIG. 3.2.

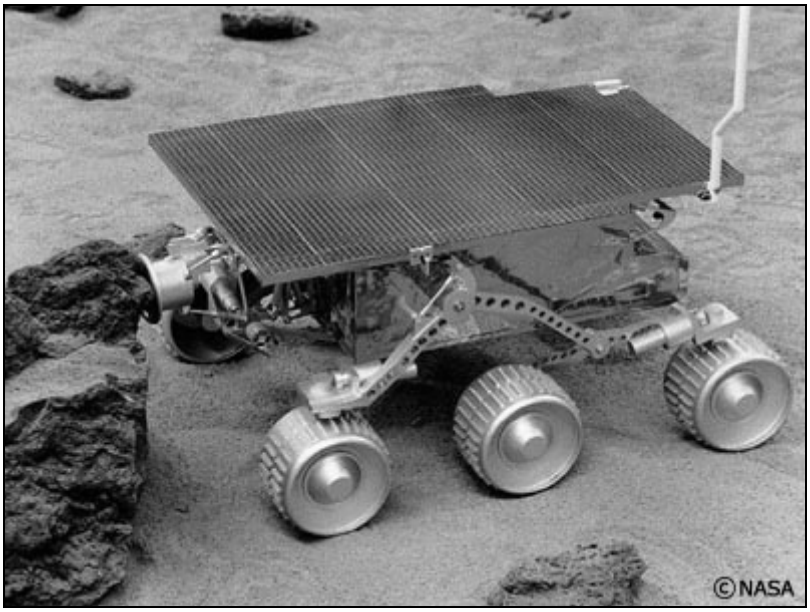

FIG. 3.2 – Robô Sojourner.

O terceiro tipo de robô é um tipo híbrido, consistindo em um robô móvel equipado com manipuladores. Um robô que se enquadra nesse tipo é o robô humanóide, cuja estrutura física se assemelha à dos humanos. Os híbridos têm a vantagem de poder aplicar seus efetuadores em locais onde os manipuladores não alcançam, mas têm a desvantagem de não possuírem a mesma rigidez de um manipulador, por não possuírem um ponto de fixação.

O campo da robótica também inclui diversos dispositivos protéticos (membros artificiais), ambientes inteligentes (como uma casa com sensores e efetuadores) e sistemas com vários corpos, nos quais a ação robótica depende da ação de um conjunto de pequenos robôs cooperativos.

Os robôs reais devem lidar com ambientes parcialmente observáveis, estocásticos, dinâmicos e contínuos. Alguns ambientes podem ser seqüenciais e multiagentes. A observabilidade parcial e o caráter estocástico são o resultado do trabalho em um mundo

grande e complexo. O robô não pode ver através de obstáculos e os comandos de movimento estão sujeitos a incertezas, devido ao deslizamento de engrenagens, à fricção, vento etc. Em um ambiente simulado, é possível utilizar algoritmos de aprendizado (como o *Q-Learning*, apresentado na subseção 3.4.4) para aprender coisas consumindo apenas algum tempo de CPU. Em um ambiente real, esse aprendizado poderia levar anos, além da possibilidade de ocorrerem danos reais ao robô devido a colisões, desgastes etc. Os sistemas robóticos práticos precisam incorporar conhecimento sobre o robô, sobre o ambiente de atuação e sobre as tarefas a serem executadas pelo robô, de forma que o robô possa aprender com rapidez a executar suas tarefas em segurança.

3.2.1 SENSORES

O robô pode perceber seu ambiente através dos sensores. Os sensores podem ser passivos ou ativos. Os sensores passivos, como a câmera, apenas "observam" o ambiente, captando sinais gerados por fontes no próprio ambiente. Os ativos, como o sonar, enviam energia ao ambiente e contam com o fato de que essa energia é refletida de volta para o sensor. Ainda segundo (RUSSEL e NORVIG, 2004), os sensores ativos tendem a fornecer mais informações que os sensores passivos, mas ao custo de maior consumo de energia e com um perigo de interferência, principalmente quando vários sensores ativos são usados ao mesmo tempo. Sejam ativos ou passivos, os sensores podem ser divididos em três tipos. Os telêmetros são sensores que medem a distância até objetos próximos e são mais comuns em robôs móveis. Os sensores de tratamento de imagens podem fornecer imagens do ambiente e, usando técnicas de visão computacional, modelos e características do ambiente. Os sensores proprioceptivos informam ao robô seu próprio estado, medindo a configuração exata de uma articulação, por exemplo.

Como o tipo de robô de maior relevância para esse trabalho é o robô móvel, em particular o VANT, são apresentados a seguir alguns dos mais importantes sensores para esse tipo de veículo. Os sensores apresentados permitem, principalmente, determinar a posição dos robôs e para onde estão direcionados dentro de seu ambiente. A partir dessas informações, fica mais fácil deslocar-se de um ponto a outro do ambiente de forma mais controlada. Outros sensores permitem que o robô seja capaz de perceber o ambiente, ou seja, detectar, localizar e até reconhecer objetos do ambiente. As subseções seguintes, baseadas em (BORENSTEIN *et al.*, 1996), descrevem os principais sensores usados em Robótica Móvel.

Sensores Inerciais

Sensores inerciais são particularmente importantes em veículos robóticos, pois fornecem informações de posicionamento, atitude e orientação, que, geralmente, são utilizadas em sistemas de controle. Os tipos mais comuns de sensores inerciais são os giroscópios, inclinômetros e acelerômetros.

Giroscópios

São sensores que fornecem a velocidade angular do veículo dentro de um determinado ambiente. Veículos aéreos geralmente necessitam de três giroscópios montados de forma a fornecerem as taxas de rotação nos três eixos.

Inclinômetros

Estes dispositivos fornecem os ângulos de inclinação em relação ao plano horizontal. Na realidade, esses dispositivos não medem diretamente os ângulos de inclinação, mas sim fornecem essas medidas baseadas em outro conjunto de informações geradas por outros sensores (acelerômetros, giroscópios, bússola).

Acelerômetros

São dispositivos que fornecem a informação de aceleração em um determinado eixo e, por esse motivo, é necessário utilizar três desses dispositivos de forma a obter as medidas nos três diferentes eixos. Os dados de aceleração, juntamente com as informações fornecidas pelos giroscópios, são utilizados nos mecanismos de navegação inercial. Esses dados, quando integrados ao longo do tempo, fornecem informações de posição, atitude e de orientação de um veículo dentro de um determinado ambiente.

Bússolas

A orientação do veículo em um determinado ambiente é um dos mais importantes parâmetros utilizados em navegação. Por esse motivo, sensores que possuam a capacidade de fornecer a orientação absoluta são extremamente importantes para a navegação de um veículo autônomo. O campo magnético da Terra é usado como referência básica de orientação, sendo necessário realizar correções, pois existe uma diferença entre o norte geográfico e o norte magnético, conhecida como declinação. A declinação também varia com o tempo e a posição geográfica

na Terra, sendo que existem mapas e gráficos apropriados que indicam tais variações para cada posição no globo terrestre.

GPS – Global Positioning System

Este sistema permite determinar a posição absoluta do veículo (latitude, longitude e altitude) em qualquer parte do globo terrestre. Apesar dos sensores inerciais poderem fornecer informações de posicionamento e orientação, os erros de integração se tornam muito significativos com o decorrer do tempo. Como os VANTs percorrem distâncias relativamente grandes, comparados aos outros tipos de veículos autônomos, a utilização de um sensor que forneça o posicionamento absoluto se torna importante para a realização de navegação. O *Navstar Global Positioning System* utiliza uma constelação de 24 satélites orbitando a terra a cada 12 horas, a uma altura de 20.186 km. Quatro satélites estão localizados em cada um dos seis planos inclinados, a 55 graus com relação ao plano equatorial da terra. A posição absoluta nas três dimensões de qualquer receptor de GPS é determinada por técnicas simples de triangulação baseadas no tempo de vôo dos sinais de rádio identificados através de um código diferenciado para cada um dos satélites. A medida exata do tempo de propagação é convertida em *pseudoranges*, que representam a distância entre o receptor e certo número de satélites em posições orbitais conhecidas. Conhecendo a distância exata entre os satélites, é possível calcular a latitude, longitude e a altitude do receptor. Os satélites enviam dois tipos de código: código C/A (*coarse acquisition*), para aplicações civis, e o código Y (criptografado), que é de uso militar apenas. Para aplicações comerciais, o código C/A possui pequenos erros de temporização deliberadamente introduzidos pela estação mestre, de forma a evitar que este sistema seja utilizado para realizar o envio de mísseis. Esse erro degrada a medida de posicionamento em cerca de 100 metros, podendo chegar até cerca de 200 metros.

GPS Diferencial

Os erros apresentados pelo GPS normal podem ser minimizados através do uso de uma prática conhecida como DGPS (*Differential GPS*). O conceito é baseado na premissa de que um segundo receptor de GPS está bem próximo (dentro de um raio de 10 km) do receptor que será utilizado, e que ambos experimentam o mesmo erro quando estão utilizando os mesmos satélites de referência. Se o segundo receptor estiver numa posição conhecida, este pode calcular o vetor erro de posição e passar esta informação para o primeiro, de forma a eliminar os efeitos do erro inserido no sinal. Esta técnica permite reduzir o erro de posição para um

raio de cerca de 5 metros (sistemas comerciais disponíveis atualmente). A partir de dezembro de 2000, o erro proposital introduzido no sinal C/A foi retirado, o que permite atualmente a utilização do sistema de GPS com uma precisão muito maior, tornando a utilização do DGPS facultativa em diversos tipos de aplicação.

Altímetro

Uma importante medida, principalmente para veículos aéreos, é a de altitude. Apesar do GPS fornecer a medida de altitude, esta está sujeita a atrasos e imprecisões (observar que a medida de altitude dada pelo GPS é a que possui o maior erro) que comprometem muito o desempenho do controle sobre o veículo aéreo. Uma das formas para se obter a altitude com uma boa resolução e sem atrasos é através do uso de um altímetro barométrico. Este dispositivo mede a variação da pressão atmosférica, que pode ser facilmente convertida para variação de altitude.

Câmera

As imagens do ambiente são importantes para os robôs móveis quando eles precisam reconhecer objetos, ou quando precisam obter um modelo do ambiente para auxiliar o sistema de navegação. As câmeras são sensores de baixo custo quando comparadas com os outros sensores. O custo pode variar, dependendo do tamanho e da qualidade da imagem que se deseja obter com a câmera. As câmeras podem capturar imagens em diferentes resoluções, a diferentes taxas de quadros por segundo. A qualidade é importante, principalmente porque, utilizando técnicas de visão estereoscópica, é possível reconstruir um modelo 3D do ambiente a partir de duas ou mais câmeras calibradas e posicionadas em pontos diferentes (SINOPOLI *et al.*, 2001). A qualidade da reconstrução pode variar de acordo com a resolução das imagens obtidas e a posição relativa entre as câmeras. Por exemplo, a visão humana consiste de dois olhos que apontam para um mesmo local alvo. Como os dois olhos estão ligeiramente afastados um do outro, o cérebro é capaz de realizar uma triangulação e determinar a distância do alvo, desde que o alvo esteja a uma distância máxima de aproximadamente 10 metros. As câmeras também possuem limitações, como a luminosidade do ambiente, que não pode ser muito baixa, apesar de existirem no mercado as câmeras noturnas. Outro problema para as câmeras são os motores, que podem ocasionar interferências e ruídos nas imagens.

3.2.2 EFETUADORES

Os efetuadores são os meios pelos quais os robôs se movem e alteram a forma de seus corpos. Para entender o projeto de efetuadores, é preciso saber o conceito de grau de liberdade (GDL). Contamos um grau de liberdade para cada direção independente em que um robô ou um de seus efetuadores pode se mover. Por exemplo, um robô rígido de movimentos livres como um UAV tem seis graus de liberdade, sendo três para sua posição (x, y, z) no espaço e três para sua orientação angular, conhecidos como *yaw*, *roll* e *pitch*. Esses seis graus de liberdade definem o "estado cinemático" ou "pose" do robô. O "estado dinâmico" de um robô inclui uma dimensão adicional para a taxa de mudança de cada dimensão cinemática.

Corpos não rígidos podem possuir graus de liberdade adicionais dentro do próprio robô, devido às suas articulações. Essas articulações podem ser de revolução – quando geram movimento de rotação – ou prismáticas – quando geram um movimento de deslizamento. Para robôs móveis, existe uma variedade de mecanismos para locomoção, incluindo rodas, esteiras, pernas e propulsores. Os robôs de tração diferencial possuem mecanismos de locomoção com acionamento independente, realizando curvas usando movimentos diferentes em cada mecanismo de locomoção, como ocorre em um tanque de guerra, por exemplo. Uma alternativa é a tração sincronizada, onde cada mecanismo de locomoção pode girar em torno de seu próprio eixo. Isso poderia ser um problema, se não fosse a restrição de que todos devem apontar para a mesma direção e mover-se na mesma velocidade. A seguir são apresentados alguns dos efetuadores mais usados em VANTs, principalmente nos sistemas de propulsão, controle de direção e de velocidade.

Propulsão

Motores a combustão interna

Os motores a combustão interna são amplamente utilizados em VANTs por apresentarem algumas vantagens sobre outros tipos de motores, porém apresentam uma série de desvantagens que também serão citadas adiante. Os motores a combustão possuem baixo peso por potência gerada, ou seja, podem ser bastante leves e compactos e ainda assim gerar uma potência considerável. Outra grande vantagem, principalmente para veículos aéreos, é que o combustível consegue fornecer uma potência maior que aquela fornecida por uma bateria com o mesmo peso. Devido a essas qualidades, esses motores são freqüentemente utilizados nos

sistemas de propulsão dos VANTs. Entretanto, alguns problemas podem impedir ou dificultar essa utilização em VANTs, como: nível elevado de ruído e vibração, que podem danificar equipamentos e a estrutura do veículo; se o motor falhar durante um vôo, pode não ser possível ligá-lo novamente, devendo deixá-lo ligado mesmo sem necessidade e consumindo combustível desnecessariamente; motores a combustão são projetados para girar em um único sentido e por esse motivo, para obter a reversão, é necessário adaptar uma caixa de engrenagens, acrescentando peso, volume e complexidade ao sistema; motores a combustão são não-lineares e possuem zonas mortas que dificultam a implementação de controladores; geração de resíduos, muitas vezes tóxicos ou que podem ser danosos aos equipamentos embarcados (corrosivos, oxidantes etc.).

Motores Elétricos DC

Motores elétricos DC também podem ser utilizados como propulsores; porém, como já foi apresentado anteriormente, baterias não apresentam uma relação "potência gerada/peso" muito favorável para ser utilizada em sistemas embarcados. Uma vantagem desses motores é que eles podem ser ligados e desligados em pleno vôo sem problemas, economizando energia. Outras vantagens são: a possibilidade de se implementar a reversão muito facilmente e a simplicidade de implementação dos controladores, se comparados com os motores a combustão. Esse tipo de motor é mais utilizado em veículos robóticos com rodas, pois estes em geral não necessitam atingir grandes velocidades, não geram ruído excessivo e nem emitem gases (o que pode ser um problema sério dentro de ambientes fechados), e o peso das baterias não chega a ser problema significativo. Alguns VANTs utilizam motores elétricos como propulsores, porém a utilização desses pode limitar o tempo de vôo devido à baixa durabilidade das baterias. Existem dois projetos de dirigíveis que procuram utilizar células fotovoltaicas para alimentar seus sistemas (KRÖPLIN, 1998). Contudo, tais sistemas devem possuir uma capacidade de carga maior para permitir o transporte de baterias e de painéis com células fotovoltaicas.

Atuação

Motores de passo

São muito utilizados quando se necessita de uma grande precisão de posicionamento. Esse tipo de motor é amplamente utilizado em sistemas robóticos, principalmente naqueles que

necessitam de um controle preciso de movimentação de câmeras de vídeo, e em sistemas de mapeamento por laser. Apesar da facilidade de se realizar a interface desse tipo de motor com computadores e microcontroladores, um grande problema deste é o seu peso elevado em relação ao torque fornecido.

Servos

Esse tipo de atuador é amplamente utilizado em modelismo e, por este motivo, existe uma grande variedade de modelos disponíveis comercialmente. Apesar de apresentar uma precisão de posicionamento baixa, ele é ideal para diversos tipos de montagem devido ao seu baixo custo, robustez, leveza e tamanho. Também possui um torque elevado em relação ao seu peso. Em geral é composto por um motor DC, um conjunto de engrenagens de redução e um circuito de controle por realimentação ligado ao eixo de saída do servo. Servos são muito utilizados para controlar a velocidade de rotação de motores a combustão e também para posicionar as superfícies de atuação (*rudder*, *elevator*, *ailerons* etc.).

3.2.3 OUTROS HARDWARES DE ROBÔS

Os sensores e os efetuadores não são os únicos hardwares necessários para a construção de um robô móvel. Um robô completo também precisa de uma fonte de energia para alimentar seus efetuadores. Apesar de o motor elétrico ser o mecanismo mais popular para atuação dos manipuladores e para a locomoção dos robôs, a atuação pneumática (que utiliza gás comprimido) e a atuação hidráulica (que utiliza fluidos pressurizados) também têm seus nichos de aplicação. A maioria dos robôs precisa ainda de algum meio de comunicação digital, como uma rede sem-fio. Uma unidade central de processamento também se faz necessária quando é preciso processar as entradas dos sensores no próprio robô. Finalmente, tem de haver uma estrutura corporal onde se possam prender todas as pequenas peças e ferragens que compõem a estrutura do robô. Além dos citados, existem outros dispositivos que podem ser usados na construção de um robô, principalmente se esse robô deve realizar algum tipo de tarefa especializada.

3.2.4 PERCEPÇÃO ROBÓTICA

Os sensores são os responsáveis pela percepção que um robô tem de seu ambiente (RUSSEL e NORVIG, 2004). Através de uma seqüência de percepções, um robô pode criar uma representação (ou modelo) do ambiente. Além dos ruídos ocasionados por motores,

ondas etc., a percepção também é dificultada pelo fato de, geralmente, o ambiente ser parcialmente observável, dinâmico e imprevisível. Para que as coisas funcionem bem na prática, é preciso que a representação ou o modelo interno do ambiente tenha pelo menos três propriedades: deve conter informações suficientes para o robô tomar as decisões corretas; sejam estruturadas de tal modo que possam ser atualizadas com eficiência; e sejam naturais, no sentido de que as variáveis internas correspondam às variáveis de estado naturais do mundo físico.

A localização é um exemplo genérico de percepção de robôs. Trata-se do problema de determinar onde os objetos estão localizados, fator fundamental para qualquer interação física bem sucedida. Esse é um dos problemas mais pervasivos da robótica. O problema se torna ainda mais complexo quando se deseja localizar muitos objetos. É quando o problema passa a ser o de mapeamento robótico, freqüentemente chamado de Localização e Mapeamento Simultâneos (LEMS, ou SLAM, de *Simultaneous Localization and Mapping*). O robô deve não apenas construir um mapa do ambiente em que se encontra, mas deve fazê-lo sem saber onde está. LEMS é um dos problemas centrais da robótica, e se torna ainda mais complexo quando o ambiente pode mudar enquanto o robô se movimenta por ele. Um dos grandes desafios da robótica é resolver o problema de LEMS com o mínimo de recursos possível, e esse também é um dos maiores desafios no projeto maior do qual esta dissertação é parte. Resolver o problema de LEMS com precisão é uma tarefa árdua, e muitas vezes se lança mão de técnicas que retornam estimativas probabilísticas dos diversos valores envolvidos em tal problema, em vez de se tentar obter soluções exatas com alto custo computacional.

Apesar de a palavra “robô” normalmente indicar uma máquina constituída de diversas partes físicas reais, muitas vezes se faz necessária uma representação virtual desta máquina. Dependendo da complexidade do contexto físico no qual o robô está inserido, uma prática indispensável para o desenvolvimento e o aprimoramento do robô é a simulação computacional. Para tal, além de modelar o comportamento do robô, é necessário lançar mão de técnicas de Computação Gráfica capazes de representar o robô e seu espaço de trabalho, que, na maioria das vezes, consiste em um espaço tridimensional. A próxima subseção apresenta os conceitos básicos da Computação Gráfica e, ainda, os fundamentos da biblioteca gráfica OpenGL, que consiste em uma API para renderização de imagens em duas e três dimensões.

3.3 COMPUTAÇÃO GRÁFICA E OPENGL

A computação gráfica é matemática e arte. A relação entre luz, tempo e movimento constitui a base desta que poderia ser classificada como uma arte tecnológica. Parece haver um consenso entre os pesquisadores de que o primeiro computador a possuir recursos gráficos de visualização de dados numéricos foi o *Whirlwind I*, desenvolvido pelo MIT em 1950, para fins militares. Após alguns anos, a década de 1990 marcou o amadurecimento da computação gráfica com algumas imagens impressionantes, como no filme *Jurassic Park*, de 1993. A popularização da computação gráfica foi impulsionada com o surgimento da biblioteca gráfica OpenGL, lançada em 1992 e largamente utilizada até os dias de hoje. Este capítulo é focado nos conceitos básicos da Computação Gráfica, bem como nos fundamentos da biblioteca gráfica OpenGL.

3.3.1 VISÃO GERAL

A computação gráfica engloba quatro grandes subáreas: a *síntese de imagens*, a *modelagem*, o *processamento de imagens* e a *análise de imagens*. A síntese de imagens considera a criação sintética das imagens, ou seja, as representações visuais de objetos criados pelo computador a partir das especificações geométricas e visuais de seus componentes. A modelagem insere-se no campo da Ciência da Computação, utilizando-se da Computação Gráfica e da Matemática Aplicada e Computacional como ferramentas para criar representações abstratas (virtuais) de Sistemas Físicos - Objetos ou Processos. O processamento de imagens considera o processamento das imagens na forma digital e suas transformações, por exemplo, para melhorar ou realçar suas características visuais. Finalmente, a análise de imagens considera as imagens digitais e as analisa para a obtenção de características desejadas, como, por exemplo, a especificação dos componentes de uma imagem a partir de sua representação visual. O mercado atual abrange áreas óbvias, como arquitetura e engenharia, e outras áreas como medicina, turismo, psicologia etc.

O contínuo avanço da computação gráfica não seria possível sem o avanço paralelo do hardware necessário para processar e apresentar as imagens, como as placas de vídeo, monitores, HMDs (*Head Mounted Display*) etc. A percepção espacial das imagens geradas é alcançada graças a estudos sobre a forma como os seres humanos enxergam tridimensionalmente os objetos do mundo. A percepção espacial humana é baseada

principalmente em algumas informações monoculares e em informações estereoscópicas, apresentadas a seguir (AZEVEDO, 2003).

Informações monoculares

As informações monoculares são aquelas inerentes à imagem plana formada na retina de cada um dos olhos. Dentre as informações monoculares, pode-se citar a noção de perspectiva, o conhecimento prévio do objeto, a oclusão, a densidade das texturas, a variação da reflexão da luz e as sombras.

Perspectiva

A noção de perspectiva linear é o resultado da aparente diminuição dos tamanhos e das distâncias entre os objetos à medida que o observador se distancia destes. Atualmente, esse recurso é largamente usado para expressar cenas tridimensionais em superfícies planas (papel, monitor de vídeo etc.).

Conhecimento Prévio do Objeto

O conhecimento prévio do tamanho de um objeto serve tanto para determinar a distância absoluta a partir do observador, quanto as distâncias relativas entre os objetos. Além disso, quando há dois ou mais objetos no mesmo campo de visão e o observador tem noção de seus tamanhos reais, o tamanho aparente serve para determinar qual deles está mais próximo ou mais distante.

Oclusão

A oclusão é responsável pela informação da posição relativa dos objetos. Este fenômeno, também chamado de interposição ou interrupção de contorno, é descrito como a obstrução da visão de um objeto por outro que está mais próximo do observador e sobre uma mesma direção de visão. Assim, quando um objeto A obstrui um objeto B, o cérebro sabe que o objeto A está mais próximo do que o objeto B.

Densidade de Texturas

Conhecido também como “gradiente de texturas”, este fenômeno visual baseia-se no fato de que muitos objetos possuem em sua aparência algum tipo de padrão com certa regularidade.

Nesse caso, à medida que os padrões aparecem mais densos e menos detalhados, mais distantes estarão do observador. As texturas também auxiliam na percepção do movimento. Se uma esfera metálica polida estiver girando, dificilmente um observador perceberá o movimento. Caso ela possua uma textura quadriculada, o movimento é facilmente percebido.

Variação da Reflexão da Luz

A mudança da intensidade da luz refletida ao longo de uma superfície de um objeto fornece informações sobre a forma e a curvatura da superfície desse objeto. Se não for gerada uma variação na cor dos pontos da superfície, a identificação do objeto pode se tornar difícil.

Sombras e Sombreamento

Este efeito é útil na determinação da posição de um objeto em relação a um piso colocado abaixo deste, ou na definição relativa entre objetos.

Informações Visuais Estereoscópicas

Como os olhos estão posicionados em lugares diferentes, cada um deles vê uma mesma cena de forma diferente. Essa diferença é chamada disparidade binocular. O cérebro usa essas diferenças para obter a distância relativa dos objetos. A estereoscopia é útil na noção da distância de objetos colocados até 10 metros do observador. Atualmente, existe uma diversidade grande de dispositivos voltados para a habilidade de perceber a profundidade com pares de imagens em estéreo. Porém, ainda não está claro o quanto a percepção de profundidade depende puramente das disparidades geométricas ou o quanto isso está relacionado à familiaridade de objetos já conhecidos.

3.3.2 OPENGL

A OpenGL (*Open Graphics Library*) (OPENGL, 2007) pode ser definida como uma interface de software (API – *Application Program Interface*) para aceleração da programação de dispositivos gráficos, com aproximadamente 120 comandos para especificação de objetos e operações necessárias para a produção de aplicações gráficas interativas 3D. Na verdade, podemos classificá-la como uma biblioteca de rotinas gráficas para modelagem 2D ou 3D. Ela é extremamente portável e rápida, possibilitando a criação de gráficos 3D com excelente qualidade visual e com bastante rapidez, uma vez que usa algoritmos consagrados e otimizados pela *Silicon Graphics*. Justamente pela sua portabilidade, não possui funções para

gerenciamento de janelas, interação com o usuário ou arquivos de entrada/saída. Cada ambiente, como, por exemplo, o *Microsoft* Windows, possui sua própria API para esse propósito.

A OpenGL não é uma linguagem de programação como C, C++, Delphi ou Java, mas sim uma poderosa e sofisticada biblioteca de códigos, para desenvolvimento de aplicações gráficas 3D de tempo real, seguindo a convenção de chamada de bibliotecas da linguagem de programação C. Isso significa que programas escritos nessa linguagem podem facilmente chamar as funções, tanto porque estas foram escritas em C, como por ser fornecido um conjunto de funções C intermediárias que chamam funções escritas em Assembler ou outra linguagem de programação. Porém, podemos utilizar também Ada, C++, Fortran, Pyton, Perl, Java, Delphi, Visual Basic e outras. Normalmente, se diz que um programa é *baseado em OpenGL* ou é uma *aplicação OpenGL*, o que significa ser escrito em uma linguagem de programação que faz chamadas a uma ou mais de suas funções.

Os principais fabricantes de PC e supercomputadores como *SGI*, *Cray Research*, *Compaq*, *Fujitsu*, *HP*, *Hitachi*, *IBM*, *Intel*, *Microsoft*, *Mitsubishi*, *NEC*, *Samsung*, *Siemens*, *Sony* e *Sun Microsystems* adotaram a OpenGL como uma biblioteca de padrão aberto para hardware gráfico.

A biblioteca OpenGL vai além do desenho de primitivas gráficas, tais como linhas e polígonos. Ela também dá suporte à iluminação, sombreamento, mapeamento de textura e transparência. A biblioteca GLU (que é parte da implementação da OpenGL) possui várias funções para modelagem, tais como superfícies quadráticas, curvas e superfícies NURBS (*Non Uniform Rational B-Splines*). Além disso, a biblioteca OpenGL executa transformações de translação, escala e rotação, através da multiplicação de matrizes com transformações cumulativas, ou seja, umas sobre as outras.

Apesar de ser uma biblioteca de programação padronizada, existem muitas implementações da OpenGL, por exemplo, para Windows e para Linux. Das implementações para PC, as mais conhecidas são as da *Silicon Graphics* e da *Microsoft*. No caso da implementação da Microsoft, são fornecidas, com suas ferramentas de programação, as bibliotecas opengl32.lib (OpenGL) e glu32.lib (utilitários OpenGL). A OpenGL possui como principais características: aceleração do hardware e do processamento geométrico, efeitos de luzes, transformações geométricas, renderizador; efeitos 3D em tempo real, como atmosfera,

anti-aliasing, sombras etc.; suporte a inovações futuras de software e hardware; estabilidade, pois vem sendo utilizada por estações avançadas e supercomputadores desde 1992; escalabilidade, podendo ser executada em supercomputadores e até em pequenos aparelhos eletrônicos.

De forma semelhante a outras APIs do Windows, os comandos da OpenGL da *Microsoft* são disponibilizados através de bibliotecas dinâmicas, conhecidas como DLLs, descritas a seguir:

- OpenGL32.dll: contém as funções padrão da OpenGL definidas pelo OpenGL *Architeture Review Board*. Essas funções são caracterizadas pelo prefixo *gl*.

- GLU32.dll: contém funções com o prefixo *glu*, para desenho de esferas, cubos, discos, cilindros etc.

- GLAux.dll: contém funções caracterizadas pelo prefixo *aux* e, embora não sejam realmente parte da especificação OpenGL, permitem desenvolver aplicações simples, independentes de hardware e sistema operacional.

- GLUT32.dll: OpenGL *Utitily Toolkit* é um sistema de gerenciamento de janelas independente de sistema operacional, escrito por (KILGARD, 1996). Todas as suas rotinas começam com o prefixo *glut*.

Adiante, são apresentadas algumas ferramentas da computação gráfica, e, quando aplicável, os comandos da OpenGL que correspondem ao uso dessas ferramentas.

3.3.3 TRANSFORMAÇÕES GEOMÉTRICAS

A habilidade de representar um objeto em várias posições no espaço é fundamental para compreender sua forma. A possibilidade de submetê-lo a diversas transformações é importante em diversas aplicações da computação gráfica. As operações lineares de rotação e translação de objetos são chamadas operações ou transformações de corpos rígidos. As transformações geométricas são também úteis em outras áreas, como na Robótica, por exemplo. A seguir, são apresentadas as principais transformações.

Transformação de Translação

Transladar significa movimentar o objeto. Translada-se um objeto deslocando todos os seus pontos, adicionando quantidades às suas coordenadas. No sistema 3D, por exemplo, o ponto pode ser movido T_x unidades em relação ao eixo x, T_y unidades em relação ao eixo y e T_z unidades em relação ao eixo z, o que pode ser descrito como uma soma de dois vetores, $P = \begin{bmatrix} x & y & z \end{bmatrix}$ e $T = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}$, resultando em $P' = \begin{bmatrix} x' & y' & z' \end{bmatrix}$, ou seja, $P' = P + T$.

A operação de translação pode ser facilmente realizada usando o comando da OpenGL *glTranslatef*(T_x, T_y, T_z), onde T_x, T_y e T_z são os parâmetros que devem ser passados com os valores de translação nos respectivos eixos coordenados x, y e z.

Transformação de Escala

Para fazer com que uma imagem definida por um conjunto de pontos mude de tamanho (ou de escala), é necessário multiplicar os valores de suas coordenadas por um fator de escala. No sistema 3D, essa operação pode ser representada na forma:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} = \begin{bmatrix} xS_x & yS_y & zS_z \end{bmatrix} \qquad \text{(EQ 3.1)}$$

Vale lembrar que, se o objeto não estiver definido em relação à origem, essa operação de multiplicação de suas coordenadas por uma matriz também fará com que o objeto translade. A operação de escalonamento pode ser facilmente realizada usando o comando da OpenGL *glScalef*(S_x, S_y, S_z), onde S_x, S_y e S_z são os parâmetros que devem ser passados com os valores de escalonamento nos respectivos eixos coordenados x, y e z.

Transformação de Rotação

Se um ponto de coordenada (x, y), distante $r = \sqrt{x^2 + y^2}$ da origem do sistema de coordenadas, for rotacionado de um ângulo θ em torno da origem, suas coordenadas, que antes eram definidas como: $x = r \cdot \cos\phi$, $y = r \cdot sen\phi$, passam a ser descritas como (x', y') dadas por

$$x' = r \cdot \cos(\theta + \phi) = r \cdot \cos\phi \cdot \cos\theta - r \cdot sen\phi \cdot sen\theta \qquad \text{(EQ 3.2)}$$

$$y' = r \cdot sen(\theta + \phi) = r \cdot sen\phi \cdot \cos\theta + r \cdot \cos\phi \cdot sen\theta \quad \text{(EQ 3.3)}$$

Isso equivale às expressões

$$x' = x \cdot \cos\theta - y \cdot sen\theta \quad \text{(EQ 3.4)}$$

$$y' = y \cdot \cos\theta + x \cdot sen\theta \quad \text{(EQ 3.5)}$$

Essas expressões podem ser descritas pela multiplicação do vetor de coordenadas do ponto $\begin{bmatrix} x & y \end{bmatrix}$ pela matriz $\begin{bmatrix} \cos\theta & sen\theta \\ -sen\theta & \cos\theta \end{bmatrix}$. Essa matriz é denominada matriz de rotação no plano *xy* por um ângulo θ. No caso de o objeto não estar definido na origem do sistema de coordenadas, a multiplicação de suas coordenadas por uma matriz de rotação também resulta em uma translação. A rotação de um objeto 3D pode ser simplificada realizando as rotações individuais sobre cada eixo, usando os ângulos de Euler. Têm-se então, em vez de uma, três matrizes de rotação, sendo uma para cada eixo.

Um giro de α graus em torno do eixo *z* muda as coordenadas conforme a operação a seguir:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & sen\alpha & 0 \\ -sen\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(EQ 3.6)}$$

Um giro de β graus em torno do eixo *x* muda as coordenadas conforme a operação a seguir:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & sen\beta \\ 0 & -sen\beta & \cos\beta \end{bmatrix} \quad \text{(EQ 3.7)}$$

Um giro de δ graus em torno do eixo *y* muda as coordenadas conforme a operação a seguir:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} \cos\delta & 0 & -sen\delta \\ 0 & 1 & 0 \\ sen\delta & 0 & \cos\delta \end{bmatrix} \quad \text{(EQ 3.8)}$$

Uma observação interessante é que essas matrizes são todas ortonormais, e uma característica interessante desse tipo de matriz é que sua inversa é igual a sua transposta. Outra observação importante a ser feita é que a ordem de aplicação das transformações afeta o produto final. A operação de rotação pode ser facilmente realizada usando o comando da OpenGL *glRotatef*(θ,

R_x, R_y, R_z), onde θ é o ângulo de rotação em graus e R_x, R_y e R_z são as coordenadas do vetor sobre o qual ocorrerá a rotação. Se R_x valer 1, e R_y e R_z valerem 0, a rotação se dará em torno do eixo x. O mesmo se aplica a R_y e R_z.

3.3.4 QUATERNIÕES

Os quaterniões foram desenvolvidos há cerca de 100 anos por William Hamilton, em um trabalho matemático sobre números complexos (VICCI, 2001). Um quaternião é uma grandeza parecida com um vetor. É composto por quatro componentes e é geralmente escrito da forma

$$q = q_0 + q_x i + q_y j + q_z k \qquad \text{(EQ 3.9)}$$

Por ser um vetor de quatro dimensões, sua visualização espacial não é possível. Contudo, é possível usar quaterniões para representar a orientação de um corpo rígido em três dimensões. Mas, há um detalhe particular quanto ao uso de quaterniões para representar a orientação de um corpo rígido. Deve-se utilizar um quaternião unitário (ou normalizado), ou seja, que satisfaça a equação

$$q_0^{\,2} + q_x^{\,2} + q_y^{\,2} + q_z^{\,2} = 1, \qquad \text{(EQ 3.10)}$$

que é análoga à representação de um vetor unitário (ou normalizado).

Uma forma alternativa de definir um quaternião, principalmente quando ele representa a orientação de um corpo, é $q = [s, v]$, onde s é um valor escalar (q_0) e v é um vetor de três dimensões ($q_x i$, $q_y j$, $q_z k$). No contexto das rotações, v representa a direção para a qual o eixo de rotação aponta. Para uma dada rotação θ, sobre um eixo arbitrário representado pelo vetor unitário u, o quaternião correspondente pode ser escrito da forma

$$q = [\cos(\theta/2), \sin(\theta/2)u] \qquad \text{(EQ 3.11)}$$

É facilmente visível que, quando se trabalha com quaterniões para representar rotações, lida-se apenas com quatro variáveis em vez das nove das matrizes de rotação. A maior parte dos trabalhos sobre simulação de corpos rígidos encontrados na literatura utiliza quaterniões para representar rotações.

3.3.5 CONCEITOS COMPLEMENTARES

Os conceitos apresentados a seguir complementam os que já foram apresentados, no intuito de facilitar a compreensão dos artifícios utilizados na construção do simulador desenvolvido como parte deste trabalho.

Projeções Geométricas

Projeções permitem a visualização bidimensional de objetos tridimensionais. Para gerar a imagem de um objeto 3D, é preciso converter as coordenadas 3D em coordenadas 2D que correspondam a uma visão do objeto a partir de uma posição específica. Para isso, devem ser considerados os seguintes elementos: plano de projeção, raio projetante e centro de projeção. O plano de projeção é a superfície onde será projetado o objeto, ou seja, onde ele será representado em 2D. Os raios de projeção são as retas que passam pelos pontos do objeto e pelo centro de projeção. Centro de projeção é o ponto fixo de onde os raios de projeção partem. Um ponto se projeta no plano de projeção quando o raio de projeção intercepta o plano de projeção. As projeções planificadas podem ser paralelas ou perspectivas, as quais possuem ainda subtipos com características peculiares. A projeção paralela ortogonal é bem usada em aplicações OpenGL e pode ser selecionada pelo comando *glOrtho*(*left*, *right*, *bottom*, *top*, *near*, *far*). Os quatro primeiros parâmetros definem, respectivamente, os limites esquerdo, direito, inferior e superior da projeção. O parâmetro *near* é a menor distância desejada para um objeto visível e *far* é a maior distância desejada para um objeto visível. Dessa forma, para que um objeto seja visível, sua distância deve ser maior do que *near* e menor do que *far*. A projeção em perspectiva também é comum em aplicações OpenGL e pode ser selecionada através do comando *glFrustrum*(*left*, *right*, *bottom*, *top*, *near*, *far*). Esse tipo de projeção utiliza seis valores para definir o volume de visão em forma de tronco de pirâmide (*frustrum*).

Câmera Virtual

Quando se fotografa uma cena do mundo real, a fotografia é uma projeção da cena em um plano, que pode ser o filme ou o sensor da câmera, dependendo de ser a câmera analógica ou digital. Da mesma forma, a imagem que se obtém de uma cena sintética depende de vários fatores. Esses fatores determinam como a imagem é projetada em um plano para formar a imagem 2D exibida no dispositivo de saída, como, por exemplo, o monitor de vídeo. Em uma

aplicação OpenGL, através de alguns parâmetros, é possível indicar a posição de uma câmera e para onde ela está direcionada. Para isso, utiliza-se o comando *gluLookAt*(*EyeX*, *EyeY*, *EyeZ*, *CenterX*, *CenterY*, *CenterZ*, *UpX*, *UpY*, *UpZ*), onde o primeiro trio de parâmetros representa a posição tridimensional da câmera; o segundo trio, do centro do objeto focado; e o terceiro trio representa o vetor que aponta para o "lado de cima" da câmera. Isso significa que a câmera possui sete graus de liberdade.

3.3.6 MODELAGEM GEOMÉTRICA

Basicamente, as técnicas de modelagem se dividem em três formas: modelagem manual, automática ou matemática. A modelagem matemática usa uma descrição matemática e algoritmos para definir um objeto, geralmente através da manipulação individual de suas partículas. Efeitos como turbulência ou proliferação de organismos microscópicos são modelados usando o método matemático. A modelagem automática é a mais sofisticada, mais rápida e poderosa, e é realizada através de equipamentos especiais, como scanners 3D, que são capazes de obter o modelo tridimensional de praticamente qualquer objeto do mundo real. A modelagem manual é o método mais fácil, barato e antigo, e utiliza basicamente as medidas de um modelo real e a intuição do modelador. A forma manual foi inicialmente usada pela indústria automobilística e aeronáutica para a concepção e teste de novos modelos.

A modelagem manual conta com o uso de algumas técnicas poderosas. O *instanciamento de primitivas*, por exemplo, permite formar um sólido a partir da combinação de vários outros sólidos primitivos com operações de transformações geométricas. A *combinação de objetos* considera interseções de dois ou mais objetos para formar outro, e também é normalmente utilizada. A *geometria sólida construtiva* (CSG – *Constructive Solid Geometry*) é semelhante à combinação de objetos, sendo que as combinações podem ser de união, interseção ou diferença. Há também as modelagens por *extrusão* e por *varredura*, que podem gerar planos utilizando duas curvas (uma denominada geratriz e outra diretriz) e um sólido a partir de um plano e uma curva. A *varredura rotacional* forma sólidos a partir da rotação de uma curva ou um plano por um determinado ângulo, sobre um vetor eixo.

A modelagem manual de primitivas em uma aplicação OpenGL é feita através do comando básico *glBegin*(*tipo*), seguido pelo fornecimento de alguns vértices através do comando *glVertex3d*(*x*, *y*, *z*). O parâmetro *tipo* do comando *glBegin* é que determina o tipo de primitiva que será construída. Os parâmetros de *glVertex3d* determinam a posição espacial do

vértice. Segue uma descrição sucinta dos valores constantes que o parâmetro *tipo* pode assumir e o que eles significam.

- GL_LINES: exibe uma linha a cada dois comandos *glVertex3d*;
- GL_LINE_STRIP: exibe uma seqüência de linhas conectando os pontos definidos por *glVertex3d*;
- GL_LINE_LOOP: exibe uma seqüência de linhas conectando os pontos definidos por *glVertex3d* e, ao final, liga o primeiro ao último ponto;
- GL_POLYGON: exibe um polígono convexo preenchido, definido por uma seqüência de chamadas a *glVertex3d*;
- GL_TRIANGLES: exibe um triângulo preenchido a cada três pontos definidos por *glVertex3d*;
- GL_TRIANGLE_STRIP: exibe uma seqüência de triângulos baseados no trio de vértices v_0, v_1, v_2, depois, v_1, v_2, v_3, então, v_2, v_3, v_4, e assim por diante;
- GL_TRIANGLE_FAN: exibe uma seqüência de triângulos conectados baseados no trio de vértices v_0, v_1, v_2, depois, v_0, v_2, v_3, então, v_0, v_3, v_4, e assim por diante;
- GL_QUADS: exibe um quadrilátero preenchido, conectando cada quatro pontos definidos por *glVertex3d*;
- GL_QUAD_STRIP: exibe uma seqüência de quadriláteros conectados a cada quatro vértices; primeiro v_0, v_1, v_3, v_2, depois, v_2, v_3, v_5, v_4, então, v_4, v_5, v_7, v_6, e assim por diante.

A biblioteca OpenGL também possui uma série de sólidos 3D que podem ser criados diretamente utilizando os seguintes comandos:

- *glutSolidCube*(*size*): desenha um cubo de aresta de tamanho *size*;
- *glutSphere*(*radius*, *slices*, *stacks*): desenha uma esfera de raio igual a *radius*;
- *glutCone*(*radius*, *height*, *slices*, *stacks*): desenha um cone de raio *radius* e de altura *height*;

- *glutTorus(innerRadius*, *outerRadius*, *nsides*, *rings*): desenha um torus (semelhante a um anel) de raios *innerRadius* e *outerRadius*.

Os parâmetros *slices* e *stacks* que aparecem em algumas funções representam o número de subdivisões em torno do eixo *z* e do eixo *y*, respectivamente. Quanto maiores esses parâmetros, mais suaves serão as quinas do sólido e maior será o processamento para formá-los. Os parâmetros *rings* e *nsides* correspondem, respectivamente, ao número de seções que serão usadas para formar o *torus* e ao número de subdivisões para cada seção.

A computação gráfica ainda oferece várias técnicas para melhorar o foto-realismo das imagens sintéticas. Efeitos de iluminação, renderização de texturas e sombreamento, são alguns dos recursos disponibilizados pela biblioteca OpenGL, que, por sua vez, lança mão de algoritmos consagrados da computação gráfica para obter tais efeitos.

Os conceitos da Computação Gráfica apresentados até aqui estão longe de esgotar o assunto, ou melhor, são apenas uma introdução aos principais conceitos utilizados para a construção do simulador desenvolvido neste trabalho. O mesmo acontece com a biblioteca OpenGL. A quantidade de funções apresentadas aqui está bem aquém das mais de 120 presentes na biblioteca. Foge ao escopo da dissertação apresentar todos esses comandos. Maiores informações sobre a OpenGL podem ser obtidas na excelente documentação disponível em *http://www.opengl.org*. Maiores detalhes sobre os conceitos inerentes à computação gráfica podem ser obtidos em (AZEVEDO e CONCI, 2003).

Em aplicações de Computação Gráfica que envolvem agentes, como jogos, simuladores etc., um tipo de tarefa comum é o deslocamento do agente entre dois pontos. Esse deslocamento pode ser feio de forma desordenada ou com uma trajetória bem definida e otimizada. A próxima subseção aborda o Aprendizado por Reforço, uma técnica de aprendizado de máquinas que pode ser usada para determinar o caminho entre dois pontos de um ambiente.

3.4 APRENDIZADO POR REFORÇO

A idéia de que se aprende através da interação com o ambiente é provavelmente a primeira a ocorrer ao se pensar sobre a natureza do aprendizado. Quando uma criança brinca ou observa algo, não há um professor lhe auxiliando explicitamente, mas a criança tem uma

conexão sensorial e motora direta com o ambiente. Exercitar essa conexão produz uma grande quantidade de informação relacionada à causa e conseqüência e sobre o que fazer para atingir certos objetivos. Durante a vida, essas interações são a principal fonte de conhecimento sobre nosso ambiente. Ao aprender, sempre há a consciência de como o ambiente reage às ações realizadas, e o objetivo é sempre influenciar o ambiente através dessas ações. O aprendizado através de interações é uma idéia básica de todas as teorias de aprendizado e inteligência. Nesta subseção, é dada uma introdução sobre os fundamentos do Aprendizado por Reforço e os elementos necessários aos seus algoritmos, baseando-se em (KAELBLING *et al.*, 1996) e (SUTTON *et al.*, 1998).

3.4.1 DEFINIÇÃO

Basicamente, Aprendizado por Reforço (AR) consiste em aprender a mapear situações em ações com o objetivo de maximizar uma recompensa que é representada através de um sinal numérico. O aprendizado não visa descobrir que ações tomar, contrapondo-se à forma supervisionada de aprendizado de máquina, mas sim descobrir que seqüência de ações maximiza a recompensa, experimentando todas as ações possíveis. As ações podem afetar não só a recompensa imediata, mas também a recompensa imediatamente posterior e todas as recompensas subseqüentes. A busca da maior recompensa através de tentativa e erro e a influência das ações nas recompensas futuras são as características principais que diferenciam o AR.

A idéia é considerar os aspectos mais importantes do problema real e um agente que interaja com o ambiente para alcançar um objetivo. Esse agente deve ser capaz de capturar o estado do ambiente através de sensores e de realizar ações que alterem o estado do ambiente. O agente também deve ter um objetivo, ou vários objetivos, em relação ao estado do ambiente. Essa formulação do problema tem a intenção de incluir somente três aspectos: estado, ação e objetivo. Em relação ao tipo de aprendizado, o AR é caracterizado por realizar o aprendizado a cada interação com o ambiente, sendo um tipo de aprendizado não-supervisionado. O Aprendizado Supervisionado (AS) caracteriza-se pelo aprendizado através de exemplos fornecidos por algum tipo de supervisor externo.

O Aprendizado Supervisionado é um importante tipo de aprendizado, mas sozinho não é adequado para a realização do aprendizado através de interações, pois, em problemas interativos, muitas vezes não é possível obter exemplos de conjuntos de treinamento que

sejam, ao mesmo tempo, corretos e representativos de todas as situações em que o agente tem de interagir. Em casos em que não está disponível um conjunto de treinamento factível, um agente deve ser capaz de aprender com a experiência própria.

Um dos desafios do AR é o dilema *exploração versus explotação*. *Exploração* significa o agente selecionar ações em que o valor da recompensa não é conhecido ou não é o máximo conhecido. *Explotação* significa o agente escolher a ação que possui a maior recompensa conhecida. O dilema é que não é possível utilizar somente a exploração ou somente a explotação sem falhar no objetivo de maximizar a recompensa.

O agente precisa selecionar uma variedade de ações e progressivamente favorecer aquelas que parecem ser melhores. Se o problema for estocástico, cada ação precisa ser selecionada muitas vezes para se conseguir uma estimativa confiável da recompensa esperada. Muitas vezes uma ação que tem uma recompensa imediata baixa pode resultar em altas recompensas futuras, ou seja, os estados atingidos com aquela ação possuem recompensas altas. Por isso, é importante haver um equilíbrio entre exploração e explotação.

3.4.2 ELEMENTOS DO APRENDIZADO POR REFORÇO

Além do agente e do ambiente, podem ser identificados quatro outros elementos em um sistema de aprendizado por reforço: uma política, uma função de transição, uma função de recompensa, uma função de valor e, opcionalmente, um modelo do ambiente.

Uma política, definida por π, consiste no modo como o agente se comporta durante o tempo. Basicamente, uma política é um mapeamento dos estados do ambiente em ações a serem selecionadas quando encontrados estes estados. Em alguns casos, a política pode ser uma simples função ou tabela. Em outros casos, a determinação da política pode envolver muito poder computacional, como um processo de busca. A política é o coração do agente, sendo que ela sozinha é suficiente para determinar o comportamento do agente. Em geral, políticas podem ser estocásticas.

A função de transição, denominada δ, define as transições entre os estados. Para um estado atual, s_t, definido para um instante de tempo t, pode-se obter uma transição para um estado s_{t+1}, escolhendo para a função de transição δ uma ação apropriada no tempo t, a_t, resultando em

$$s_{t+1} = \delta(s_t, a_t) \quad \text{(EQ 3.12)}$$

Em muitos problemas reais não se tem conhecimento *a priori* desta função, mas somente em tempo real, tornando, desta forma, o processo de aprendizado totalmente dinâmico e desprovido de qualquer forma de planejamento. Nos problemas onde é possível o conhecimento *a priori* da função de transição, é factível a idealização de um modelo mental ou modelo teórico do ambiente por parte do agente, permitindo a realização de alguma forma de planejamento que possibilitará a aceleração do processo de aprendizado.

Uma função de recompensa *r* define o objetivo do problema de AR. Basicamente, ela mapeia os estados (ou pares estado-ação) do ambiente em um único número que é a recompensa, indicando a atratividade do estado. O único objetivo do agente é maximizar a recompensa total que é recebida ao longo do tempo. A função de recompensa define quais eventos são bons e quais são ruins para o agente. Em um sistema biológico, por exemplo, poderia ser apropriado identificar as recompensas como prazer e dor. Elas são características imediatas e bem definidas do problema. Como tal, a função de recompensa deve ser necessariamente fixa. No entanto, esta pode servir como base para a alteração da política.

Enquanto a função de recompensa indica qual ação é imediatamente mais vantajosa, a função de valor, indicada por V^{π}, especifica o valor de um determinado estado em longo prazo. Basicamente, o valor de um estado é o total de recompensas que um agente prevê acumular, partindo daquele estado até atingir o objetivo. Enquanto a função de recompensa determina o valor intrínseco de atratividade imediata de um estado, a função de valor indica a atratividade a longo prazo dos estados, levando em conta os estados mais prováveis de serem selecionados mais adiante e a respectiva recompensa desses estados. Essa função é importante devido ao fato de poder haver estados que têm uma recompensa baixa, seguidos de vários estados que possuem recompensas altas, elevando a função de valor dos estados anteriores significativamente. O contrário também é plausível, ou seja, estados com uma recompensa alta, mas que são seguidos de estados com uma recompensa baixa, e, portanto, possuem valor baixo, sendo pouco atrativos a longo prazo.

Valores, como previsões das recompensas futuras são, de certa forma, secundários em relação às recompensas. Isso porque, sem as recompensas, não existem valores, e o único propósito de se estimar valores é conseguir o máximo de recompensas possíveis. Entretanto, é com os valores que se deve estar mais preocupado na hora de tomar decisões e avaliá-las, pois

a seleção de ações é feita com base no julgamento de valores. Recompensas são fáceis de se determinar, pois são dadas diretamente pelo ambiente. Valores são bem mais difíceis de serem determinados que as recompensas, pois têm de ser estimados a cada seqüência de observações que um agente faz. Assim, eles podem ser corrigidos e a sua estimativa se torna mais precisa. De fato, o componente mais importante de um algoritmo de aprendizado por reforço é um método eficiente de se estimar valores, ou seja, uma função de valor eficiente.

Um quinto componente (opcional) de alguns sistemas de aprendizado por reforço é o modelo do ambiente. Seria algo que simulasse o comportamento de um ambiente. Por exemplo, dado um estado e uma ação, o modelo seria utilizado para predizer o próximo estado e a próxima recompensa. Os modelos são utilizados para a realização de planejamento, ou seja, para a implementação de algum processo discriminante de escolha, relacionado à seleção de uma ação, considerando possíveis situações futuras, mas sem que elas realmente aconteçam.

3.4.3 RETORNO AVALIATIVO

Há uma característica importante que distingue o AR de outro tipo de aprendizado. É que se usam as informações de treinamento para avaliar as ações que podem ser tomadas em vez de instruir qual ação deve ser tomada. Isto quer dizer que seu retorno é avaliativo e não, instrutivo. Devido a isso é que há a necessidade de uma exploração contínua, ou seja, de uma busca explícita através de tentativa e erro, com o objetivo de melhorar a avaliação. Na essência, o retorno avaliativo indica a qualidade da ação tomada, mas não se é a melhor ou a pior ação possível. O retorno instrutivo, por outro lado, indica a ação correta a tomar e forma a base do Aprendizado Supervisionado. Nas suas formas puras, os dois tipos de retorno supracitados são totalmente distintos: o retorno avaliativo depende inteiramente da ação tomada, enquanto o retorno instrutivo independe da ação tomada. Existem ainda casos em que esses dois tipos de retorno podem ser combinados.

Métodos Ação-Valor

Inicialmente, são descritos alguns algoritmos simples para estimar o valor das ações e para selecionar uma ação usando o valor das estimativas. Convenciona-se o valor real da ação a como $Q^*(a)$ e o valor estimado, depois de t decisões, como $Q_t(a)$. O valor real de uma ação é a recompensa média obtida uma vez que a ação é selecionada. Uma maneira natural de

estimar o valor real de uma ação é calcular a média das recompensas recebidas quando da seleção dessa ação. Para uma ação a, após t decisões em que a foi selecionada k vezes, obtendo as recompensas r_1, r_2, ..., r_k, o valor da ação a pode ser obtido através da equação

$$Q_t(a) = \frac{r_1 + r_2 + ... + r_k}{k} \qquad \text{(EQ 3.13)}$$

Entretanto, se $k = 0$ definimos o valor estimado como um valor padrão, por exemplo, $Q_t(a) = 0$. Como $k \rightarrow \infty$, pela lei dos grandes números $Q_t(a)$ converge para $Q^*(a)$. Esse método é chamado de método da média das amostras, pois cada estimativa é uma média simples das amostras das recompensas relevantes.

A regra mais simples para a seleção de ações é selecionar a ação com o maior valor estimado. Por exemplo, na decisão t, para selecionar uma das ações gulosas (com o maior valor estimado) a_t^*, tem-se que $Q_{t-1}(a_t^*) = \max Q_{t-1}(a)$. Esse método, chamado guloso, sempre usa o conhecimento adquirido para maximizar a recompensa imediata. Não é gasto nenhum tempo experimentando ações aparentemente inferiores com o objetivo de descobrir ações possivelmente melhores.

Uma alternativa simples a essa regra é selecionar ações gulosas a maioria do tempo, mas ocasionalmente selecionar uma ação aleatoriamente, com uma probabilidade ϵ. Essa seleção deve ser uniforme e independente do valor estimado de cada ação. Esses métodos são chamados de métodos ϵ-gulosos. Uma vantagem desses métodos é que, à medida que o número de decisões aumenta, todas as ações serão selecionadas um número infinito de vezes, garantindo que k tenda a infinito para todas as ações a e o valor esperado $Q_t(a)$ convirja para o valor real $Q^*(a)$. Isso implica que a probabilidade de selecionar a ação realmente ótima convirja para $1 - \epsilon$, ou seja, quase exata. Essa garantia, entretanto, não é total, e diz pouco sobre a efetividade prática desses métodos.

A vantagem do método ϵ-guloso em relação ao guloso depende muito do problema. Por exemplo, no caso de grande variância das recompensas, é necessário mais exploração para se encontrar a ação ótima, e os métodos ϵ-gulosos podem se comportar ainda melhor em relação ao método guloso. Por outro lado, se a variância for zero, então o método guloso pode determinar o real valor para cada ação depois de selecionar a ação uma única vez. Nesse caso,

o método guloso pode ter uma melhor atuação, por determinar mais rapidamente as ações ótimas e, a partir de então, não realizar mais a exploração.

Mas, mesmo no caso determinístico (variância = 0), há uma grande vantagem em explorar, pois algumas recompensas tomadas como ótimas podem não o ser. Por exemplo, suponha que o problema seja não estacionário, ou seja, que o real valor das ações se altere ao longo do tempo. Nesse caso a exploração é necessária, mesmo sendo o problema determinístico, pois uma ação não gulosa pode se tornar melhor do que a ação gulosa. Casos em que o valor das ações sofre alteração ao longo do tempo são os mais comumente encontrados em aplicações de AR, principalmente se o ambiente de interação do aprendiz é dinâmico. Mesmo no caso estacionário e determinístico, o valor esperado das ações pode se alterar ao longo do tempo devido ao próprio processo de aprendizado. Isso mostra como os problemas de Aprendizado por Reforço têm uma grande necessidade de equilíbrio entre exploração e explotação.

3.4.4 ALGORITMO Q-LEARNING

A tarefa de aprendizado de um agente que interage com um ambiente pode ser definida como determinar uma política ótima que maximize os valores das recompensas em longo prazo e estabeleça uma estratégia de controle que determine uma seqüência de ações, ou seja, a partir de um conjunto de políticas $\pi : S \rightarrow A$, responsáveis pelo mapeamento de um conjunto de estados S, em um conjunto de ações A, o agente deseja determinar uma política ótima π^*, definida como

$$\pi^* \equiv Max_{\pi} \left\{ V^{\pi}(s), \forall s \right\}. \qquad \text{(EQ 3.14)}$$

A função valor V^{π} é computada de forma acumulativa para um determinado estado s_t, a partir da soma ponderada dos retornos estabelecidos em função de uma política π, considerando um fator de desconto γ e uma função de retorno r, para um determinado horizonte, que pode ser finito ou infinito. Portanto, define-se

$$V^{\pi} \equiv r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots, \qquad \text{(EQ 3.15)}$$

para $0 \leq \gamma < 1$, sendo $r_t = r(s_t, a_t)$ a recompensa devido à ação a_t no estado s_t.

Esse problema, relacionado ao aprendizado de uma política, não pode ser tratado sob a ótica do Aprendizado Supervisionado, já que os pares do conjunto de treinamento, definidos como (s, a) = π(s), não são previamente conhecidos. O que se tem, na verdade, é uma seqüência de retornos, proveniente da transição de estados, resultantes da execução das ações por parte do agente. Isso caracteriza, portanto, um problema de atribuição de crédito temporal.

Tradicionalmente, quando se tem um conhecimento prévio de todo espaço de estados do problema, envolvendo todos os possíveis estados e suas respectivas transições e um conhecimento completo da função de recompensa ou de retorno para cada transição, a definição de uma política ótima, em casos determinísticos, consiste na solução de um problema de programação dinâmica.

Solução Aproximada e Função Q

Considerando o critério proposto por (BELLMAN, 1957), a escolha de uma ação ótima a^* no estado s está relacionada à maximização de um valor equivalente à soma do retorno imediato produzido pela ação escolhida com os valores das recompensas futuras, a partir do estado sucessor, associada ao fator de desconto γ. Portanto,

$$\pi^*(s) = ArgMax_a \{r(s,a) + \gamma \cdot V^*(\delta(s,a))\}. \quad \text{(EQ 3.16)}$$

Para que essa formulação seja empregada, é necessário que o agente aprenda a função de valor V^*, o que exige o conhecimento prévio da função de transição δ e da função de recompensa r para todas as transições possíveis, o que nem sempre pode ser realizado. Para resolver esse problema, (WATKINS, 1989) propõe o aprendizado de uma função alternativa, conhecida como função Q, que consiste em uma função de avaliação definida na forma

$$Q(s,a) \equiv r(s,a) + \gamma \cdot V^*(\delta(s,a)). \quad \text{(EQ 3.17)}$$

Considerando $\pi^*(s) = ArgMax_a Q(s,a)$ e $V^*(s) = Max_{a'} Q(s,a')$, pode-se definir a função Q na forma recursiva como

$$Q(s,a) \equiv r(s,a) + \gamma \cdot Max_{a'} Q(\delta(s,a),a'). \quad \text{(EQ 3.18)}$$

É importante observar que o aprendizado da função Q torna desnecessário o aprendizado da função valor V^* e, também, do conhecimento *a priori* das funções de recompensa r e de transição δ. Em uma forma de treinamento on-line, os valores provenientes dessas funções são

obtidos diretamente da resposta do ambiente. Para estimar a função *Q*, (WATKINS, 1989) propôs um algoritmo de treinamento baseado em correções por sucessivas diferenças temporais conhecido como *Q-Learning*. Nesse sentido, define-se uma função de aproximação *Q*, que é atualizada de forma iterativa como

$$Q(s,a) \leftarrow r(s,a) + \gamma \cdot Max_{a'} Q(s',a'), \qquad \text{(EQ 3.19)}$$

onde *s'* = δ(*s*, *a*), ou seja, no estado *s*, o agente escolhe uma ação *a* que maximiza a sua escolha, considerando os valores aproximados da função *Q*, entre todas as alternativas possíveis. Dessa forma, os valores mais recentes da função *Q* para o estado sucessor *s'* são utilizados para a atualização do valor da função *Q* para o estado *s*. Esse algoritmo pode ser estendido para situações de não determinismo, onde as funções de recompensa e de transição possuem um caráter aleatório.

Q-Learning Incremental

A atualização dos valores da função *Q*, considerando somente o efeito da ação e a conseqüente recompensa associada, implementa um processo de aprendizado baseado na adoção de um critério de correção por retorno ou diferença imediata.

Entretanto, é possível a atualização dos valores da função *Q*, considerando o efeito de um conjunto de ações antecessoras associadas às suas respectivas recompensas, com a utilização do fator de desconto λ, no sentido de incrementar ou tornar mais influentes as decisões tomadas mais recentemente. Esse processo, que também pode ser definido como um algoritmo de aprendizado de múltiplos passos, tem uma relação direta com o método mais geral de diferenças temporais, conhecido como *TD*(λ).

Na nova implementação, a única diferença em relação ao algoritmo clássico do *Q-Learning* – que atualiza um único estado por vez baseado no retorno imediato – está na forma de atualização dos valores da função *Q*. Considerando o valor da taxa de aprendizagem igual a 1 (γ = 1) e a correção em dois passos, tem-se

$$Q(x,a) = r(x,a) + \gamma \cdot \left(Max_{a'} Q(y,a') + \gamma \cdot Max_{a''} Q(z,a'')\right), \qquad \text{(EQ 3.20)}$$

para duas transições sucessivas, representadas pelas funções de transição $z = \delta(y,a')$ e $y = \delta(x,a)$. A exemplo do método *TD*(λ), pode-se considerar a existência de um número

variável de passos no processo de correção da função. Neste caso, o valor de cada estado fica ponderado pela $n^{ésima}$ potência da taxa de desconto, ou seja, multiplica-se por γ^n, considerando n o número de transições anteriores observadas.

Uma alternativa para melhorar a eficiência do algoritmo *Q-Learning* consiste na atualização simultânea de vários estados para a mesma ação escolhida, desde que possuam algum grau de similaridade com o estado atual. Nesse caso, é necessária a implementação de um processo de *matching* ou de reconhecimento através algum tipo de técnica não supervisionada, como, por exemplo, algoritmos de análise de *cluster* ou de memória associativa.

No Capítulo 6, é apresentado um sistema que estende o simulador para determinar o caminho mínimo entre dois pontos do ambiente de atuação do dirigível. O algoritmo clássico A* da IA é utilizado para ambientes estruturados e o algoritmo *Q-Learning* é utilizado para ambientes semi-estruturados e desestruturados.

3.5 CONSIDERAÇÕES SOBRE O CAPÍTULO

Nesse capítulo foram apresentados alguns conceitos fundamentais para a compreensão dos problemas e soluções abordados pela dissertação. Não cabe, porém, a elucidação de alguns conceitos básicos, geralmente de conhecimento comum às pessoas da área de engenharia e computação. Os conceitos supracitados foram transcritos de forma adaptada às necessidades da dissertação, podendo diferir sucintamente, mas não em sentido, dos conceitos originais que serviram como fonte.

O capítulo seguinte entra mais especificamente no tema da dissertação, apresentando o modelo matemático do dirigível. É apresentado o modelo matemático geral desenvolvido por (GOMES e RAMOS, 1998), bem como uma explicação mais detalhada de cada termo do modelo e as respectivas mudanças necessárias para o caso particular da abordagem do presente trabalho.

4 O MODELO MATEMÁTICO DO DIRIGÍVEL

A modelagem matemática é de fundamental importância para a simulação computacional de sistemas de controle não-lineares. O principal objetivo da modelagem é prover, em uma forma adequada e de fácil compreensão, dados de saída do sistema para os estudos que se façam necessários. Obviamente, a obtenção de um modelo preciso não deve consumir mais recursos do que realizar ensaios com o modelo real. O modelo deve ter a precisão requerida, mas deve ser suficientemente simples para permitir a determinação das suas saídas, com os meios disponíveis, em tempos adequados (compromisso simplicidade x precisão). Este capítulo apresenta o modelo matemático do dirigível, bem como uma breve descrição dos termos que o compõem. As alterações no modelo para uso e implementação no presente trabalho também são apresentadas.

4.1 INTRODUÇÃO

Quando se trata de veículos aéreos em geral (e dirigíveis em particular), é necessário se dispor de modelos razoavelmente complexos cuja determinação constitui, por si só, tema de pesquisa que extrapola os principais objetivos desta dissertação. Optou-se então pelo uso e adaptação do modelo desenvolvido por (GOMES e RAMOS, 1998), posteriormente estendido para o dirigível AS800 por (RAMOS, 2002). O modelo é reconhecido pelas comunidades acadêmica e industrial, salientando, antecipadamente, a presença de aproximações nos processos associados às necessárias mudanças de escala entre as aeronaves tratadas por (GOMES e RAMOS, 1998) e no âmbito da presente dissertação, que trata de aeronaves de menor porte.

No modelo matemático obtido por (GOMES e RAMOS, 1998), as características aerodinâmicas do dirigível foram obtidas a partir de um extenso conjunto de ensaios em túnel de vento. Numa abordagem alternativa, apresentada por (DE LAURIER e EVANS, 1981) e (CHELLI *et al.*, 2001), a obtenção dessas características é baseada no uso de técnicas de dinâmica dos fluidos computacional (CFD, do Inglês *Computational Fluid Dynamics*). Segundo (KHOURY e GILLETT, 1999), o modelo de (GOMES e RAMOS, 1998) constitui um dos mais completos modelos matemáticos de dirigível disponíveis na atualidade.

Neste capítulo, serão apresentados detalhes do modelo matemático e dos princípios básicos de operação de dirigíveis. O modelo apresentado está com algumas adaptações que objetivam abordar alguns itens não abordados por (GOMES e RAMOS, 1998), como o propulsor de cauda, que aumenta a manobrabilidade do dirigível. Assim como em (RAMOS, 2002), a força do vento também é considerada, representada por um vetor que determina a direção e a velocidade do vento e que varia de forma aleatória. Alguns termos do modelo foram suprimidos na implementação e serão comentados ao longo do capítulo.

4.2 PRINCÍPIOS BÁSICOS DE OPERAÇÃO DE DIRIGÍVEIS

A movimentação de todo corpo está diretamente relacionada com as forças que atuam sobre esse corpo. As forças podem ser exercidas de forma deliberada ou não. As subseções a seguir descrevem as forças que atuam em um dirigível, como as forças aerostática, aerodinâmica, dinâmica e, também, as forças exercidas deliberadamente, como a propulsão.

4.2.1 FORÇAS AEROSTÁTICAS

A sustentação aerostática é a principal força atuante em um dirigível, tendo em vista que é ela que sustenta o dirigível pairando no ar (KHOURY e GILLETT, 1999). Essa força é chamada de força aerostática por ser uma sustentação que é independente da velocidade de navegação, em oposição à sustentação aerodinâmica, presente nos aviões, que depende da velocidade de navegação. O princípio de Arquimedes explica a sustentação aerostática, na qual a força de empuxo F exercida num corpo é proporcional ao volume (vol) de ar deslocado por este, e também proporcional à diferença de densidade entre o ar (ρ_{ar}) no exterior e o gás ($\rho_{gás}$) no interior do envelope do dirigível, ou seja, $F = vol * (\rho_{ar} - \rho_{gás})$, dado que para o gás hélio tem-se ρ=1.06 Kg/m^3 e para o ar ρ=1,225 Kg/m^3, ambos sob condições ISA a nível do mar. Apesar de mais caro, o gás hélio é o mais usado, haja vista os riscos de explosão apresentados pelo hidrogênio.

A pressão externa exerce uma influência sobre a forma dos dirigíveis quando estes são não-rígidos, ou seja, quando não possuem uma estrutura rígida que envolve o envelope. À medida que um dirigível sobe na atmosfera, a pressão externa diminui, permitindo a expansão do gás interno e, conseqüentemente, aumentando a tensão no material do envelope, podendo

chegar a situações que colocariam em risco a resistência do envelope. O oposto ocorre nas descidas.

O uso de balonetes, que são bolsas de ar dentro do envelope principal que contém o gás de sustentação, soluciona o problema pela expulsão/admissão de ar de/para os balonetes. A presença de balonetes também é importante quando se trata de dirigíveis que perdem massa em vôo (com queima de combustível, por exemplo). A opção de não utilizar balonetes incorre em uma altitude bem limitada de vôo, que por sua vez varia de acordo com o tamanho do compartimento de armazenamento de gás e com o peso da carga embarcada no dirigível. Essa altitude limite é denominada altitude de pressão, e usualmente é a maior altitude em que um dirigível pode operar. A FIG. 4.1 (RAMOS, 2002) mostra a operação de um dirigível que faz uso de balonetes.

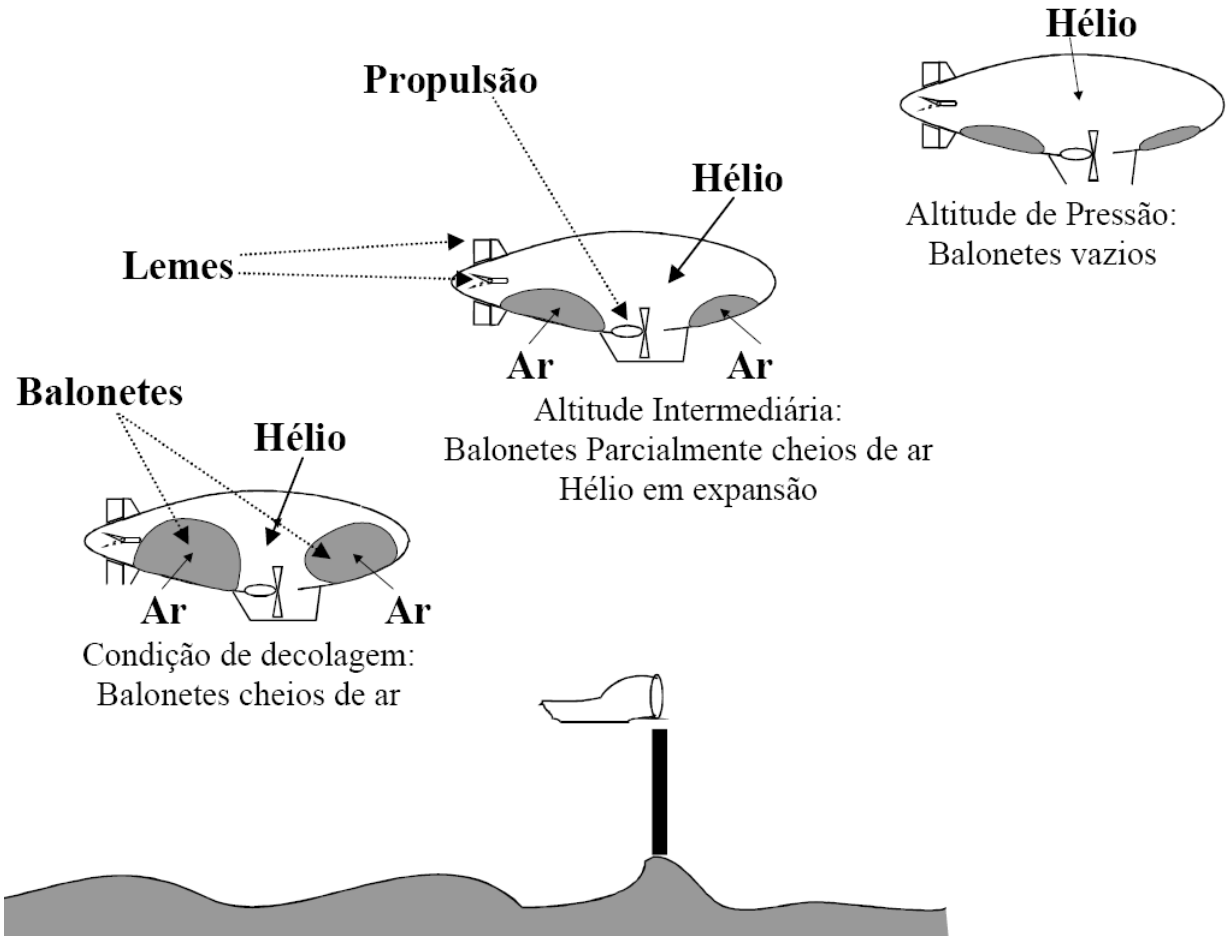

FIG. 4.1 – Operação dos balonetes em um dirigível não rígido (RAMOS, 2002).

4.2.2 FORÇAS AERODINÂMICAS E DINÂMICAS

Quando os dirigíveis se movimentam, eles também estão sujeitos a forças aerodinâmicas, distinguindo-se as seguintes forças e torques: arrasto aerodinâmico, sustentação aerodinâmica e a ação das superfícies de atuação aerodinâmicas (lemes ou profundores).

Os principais componentes das forças dinâmicas no dirigível são os associados às massas e inércias virtuais. Eles aparecem de forma significativa em veículos que flutuam num fluido, como submarinos e dirigíveis. Nesses veículos, a massa do fluido por eles deslocada é da ordem de grandeza da massa do próprio veículo. É devido à energia requerida para movimentar o fluido em torno do próprio veículo, quando este se move, que aparecem os efeitos de massas e inércias virtuais. A conseqüência prática disso é que o veículo aparenta ter massa e inércia maiores do que as que ele realmente possui. Em (KHOURY e GILLETT, 1999), encontra-se o desenvolvimento teórico do assunto e é mostrado que, para maioria dos dirigíveis, as massas e inércias virtuais são da ordem de grandeza das massas e inércias do próprio veículo quando em repouso.

4.2.3 FORÇAS DE PROPULSÃO

Geralmente, os dirigíveis de grande porte geram suas forças de propulsão através da rotação de hélices acionadas por motores à combustão. Já os dirigíveis de pequeno porte, também utilizam hélices, mas acionadas por motores elétricos ou motores à combustão baseados em metanol ou gasolina. Quando são utilizados motores elétricos, a energia provém de baterias ou de células solares. Em dirigíveis de pequeno porte, o uso de baterias oferece algumas vantagens. Uma delas é que o acionamento do motor é feito de forma mais rápida e silenciosa. Além disso, o uso de baterias permite que o motor fique totalmente parado, sem consumir energia, podendo ser colocado em operação a qualquer instante.

Para facilitar a decolagem e a aterrissagem, muitos dirigíveis são providos de propulsão vetorizável, isto é, o ângulo entre a direção dos eixos de rotação das hélices e a horizontal é variável. Eventualmente, propulsores de popa e/ou proa são montados perpendicularmente ao eixo longitudinal do veículo, para oferecer manobrabilidade adicional em baixas velocidades, como é o caso dos dirigíveis considerados neste trabalho. A FIG. 4.2 mostra os principais componentes de atuação dinâmica no dirigível considerado em (RAMOS, 2002).

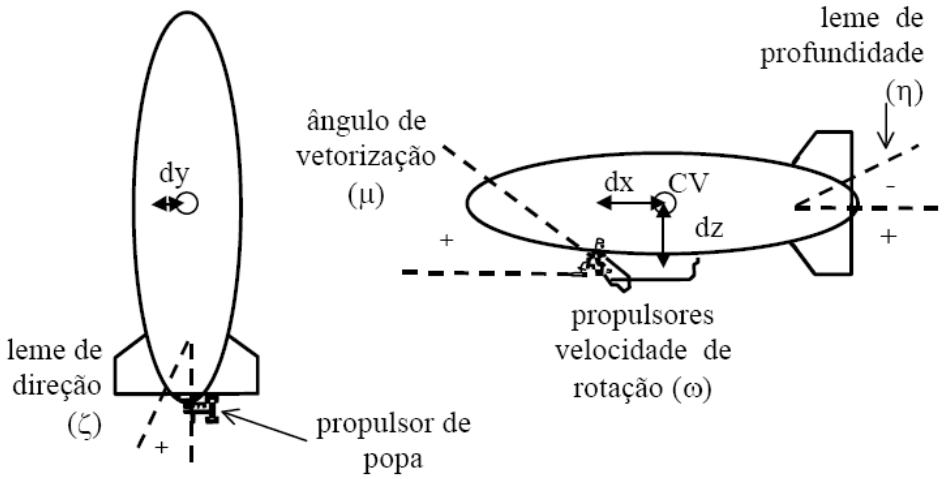

FIG. 4.2 – Componentes principais para atuação em um dirigível (RAMOS, 2002).

4.2.4 SUPERFÍCIES AERODINÂMICAS

As superfícies aerodinâmicas principais presentes em dirigíveis são os lemes de direção e de profundidade, geralmente colocados na cauda. Essas superfícies de atuação podem possuir uma configuração em "+" ou "x". Esses lemes podem atuar como estabilizadores ou como auxiliares em manobras. Quando o vento passa pela superfície do leme, um momento de força é gerado de acordo com o ângulo de ataque do respectivo leme. Caso o leme seja fixo, ele atua praticamente como um estabilizador, apenas ajudando a suavizar o movimento angular contínuo provocado pelas forças atuantes no dirigível. Caso ele possa ter seu ângulo de ataque alterado de forma independente do dirigível, através de motores, ele pode auxiliar na execução de manobras. Como um dos requisitos do sistema de dirigíveis abordado pelo presente trabalho é o baixo consumo de energia, os lemes são fixos, descartando o uso de motores adicionais.

4.3 OS TERMOS DO MODELO DINÂMICO

O modelo dinâmico do dirigível, em seis graus de liberdade, referenciado ao sistema de coordenadas no seu corpo, é dado por (GOMES e RAMOS, 1998)

$$M \cdot a = F = F_d(x) + A(x) + G + P, \qquad \text{(EQ 4.1)}$$

onde: M é a matriz de massa; a é o vetor de acelerações lineares e angulares, que pode ser obtido através da primeira derivada do vetor de velocidades lineares e angulares x; F_d é o vetor de forças e torques dinâmicos; A é o vetor de forças e torques aerodinâmicos; G é o

vetor de forças e torques gravitacionais; e P é o vetor de forças e torques de propulsão. Cada um dos componentes do modelo é descrito a seguir.

4.3.1 VETOR VELOCIDADE

O vetor velocidade x contém as três velocidades lineares [u, v, w] e as três velocidades angulares [p, q, r], todas escritas com respeito ao sistema de coordenadas fixado (SCL – Sistema de Coordenadas Local) no corpo do dirigível, sendo $x^{\mathrm{T}} = [u, v, w, p, q, r]$. O vetor aceleração a é obtido derivando-se o vetor x.

Para o desenvolvimento de estratégias de controle e navegação, bem como de simulação do modelo, as velocidades são frequentemente transformadas do sistema de coordenadas do dirigível para um sistema de coordenadas inercial (SCA – Sistema de Coordenadas do Ambiente, fixo em um ponto do ambiente), no qual o vetor de velocidades é denotado por

$$x_i^{\ T} = \left[u_N, v_L, w_A, p_N, q_L, r_A\right], \tag{EQ 4.2}$$

respectivamente associados às direções Norte, Leste e "Acima" (termos indexados por N, L e A).

4.3.2 MATRIZ DE MASSA

A matriz de massa, M, incorpora todas as massas e inércias do dirigível, incluindo os termos de massas e inércias virtuais – associados ao fato de estar se tratando de um veículo flutuante – mais a massa variável dos balonetes. Para o dirigível aqui considerado, a matriz de massa corresponde ao Tensor de Inércia (HIBELER, 2005), visto que as massas e inércias virtuais são desprezadas e que não são utilizados balonetes, por se tratar inicialmente de veículos de pequeno porte para vôo em baixa altitude.

4.3.3 VETOR DE FORÇAS AERODINÂMICAS

O vetor de forças aerodinâmicas A contém os termos aerodinâmicos do modelo, oriundos do corpo do dirigível e das superfícies de controle aerodinâmico. Ele é calculado a partir dos coeficientes aerodinâmicos de sustentação e de arrasto (respectivamentre, c_{L} e c_{D}, do inglês *lift* e *drag*), forças laterais (c_{Y}), dos momentos de arfagem ou *pitching* (c_{m}), guinada ou *yawing* (c_{n}) e rolagem ou *rolling* (c_{l}), como

$$A = F(c_i) = \left[A_X, A_Y, A_Z, A_L, A_M, A_N\right], \tag{EQ 4.3}$$

onde c_i são os coeficientes já citados; A_X, A_Y e A_Z são as forças de arrasto, lateral e de sustentação; A_L, A_M e A_N são os momentos de rolagem, arfagem e guinada respectivamente.

Os coeficientes c_i seguem a convenção padrão para dirigíveis (RAMOS, 2002), sendo

$$\begin{aligned} c_D &= A_X / \left(0.5 \cdot \rho \cdot v_{ar\_tot}{}^2 \cdot V^{2/3}\right) \\ c_Y &= A_Y / \left(0.5 \cdot \rho \cdot v_{ar\_tot}{}^2 \cdot V^{2/3}\right) \\ c_L &= A_Z / \left(0.5 \cdot \rho \cdot v_{ar\_tot}{}^2 \cdot V^{2/3}\right) \\ c_l &= A_L / \left(0.5 \cdot \rho \cdot v_{ar\_tot}{}^2 \cdot V\right) \\ c_m &= A_M / \left(0.5 \cdot \rho \cdot v_{ar\_tot}{}^2 \cdot V\right) \\ c_n &= A_N / \left(0.5 \cdot \rho \cdot v_{ar\_tot}{}^2 \cdot V\right) \end{aligned} , \qquad \text{(EQ 4.4)}$$

onde V é o volume do dirigível (em m^3), v_{ar_tot} é a velocidade relativa ao ar (em m/s) e ρ é a densidade do ar (em Kg/m^3).

Esses coeficientes aerodinâmicos podem ser obtidos por medidas diretas em túnel de vento, a partir das características geométricas do veículo (UPSON e KLIKOFF, 1925) ou a partir das derivadas de estabilidade aerodinâmicas. Para o dirigível de (GOMES, 1990), eles foram determinados diretamente por extensos testes em túnel de vento, realizados para ângulos de incidência de vento na faixa de –30º a +30º.

No presente trabalho, como não se dispunha de tais coeficientes, são utilizados coeficientes aproximados, obtidos de sólidos geométricos com forma semelhante a do dirigível, como elipsóides.

4.3.4 VETOR DE GRAVIDADE E FLUTUAÇÃO

O vetor G representa a diferença entre os vetores associados ao peso do dirigível, W, e a força de flutuação, B, agindo para cima, simplificado com base na simetria do veículo com relação ao plano XZ (GOMES, 1990). A equação fica

$$G = W - B , \qquad \text{(EQ 4.5)}$$

onde W é o peso do dirigível agindo no centro de gravidade e B é a força de flutuação agindo no centro volumétrico do envelope que contém o gás de flutuação.

4.3.5 MODELO DA FORÇA DE PROPULSÃO

O vetor P contém os termos de força e torque no sistema de coordenadas do dirigível. Assim, em termos de simulação, o vetor P, de propulsão, é determinado através dos seguintes passos:

1) Define-se a velocidade de rotação dos motores como uma percentagem da velocidade máxima permitida (sendo que nesta situação não se considera nem a dinâmica nem a resposta do motor de combustão);

2) Obtém-se os coeficientes de propulsão c_T de cada propulsor;

3) Calculam-se as forças de propulsão a partir dos coeficientes de propulsão c_T de cada propulsor, utilizando a EQ 4.6, onde ρ é a densidade do ar em Kg/m^3, n é a velocidade de rotação (em rotações por segundo) e D é o diâmetro da hélice (em metros).

$$P = c_T \cdot \rho \cdot n^2 \cdot D^4 . \qquad \text{(EQ 4.6)}$$

4.3.6 O EFEITO DO VENTO

Assim como na contribuição de (RAMOS, 2002) ao modelo de (GOMES, 1990), no simulador desenvolvido no presente trabalho foi introduzido o efeito do vento, que é a principal perturbação que afeta o dirigível em vôo e, por conseguinte, influencia no projeto do sistema de controle e navegação do veículo.

Usualmente, o vento é definido no referencial SCA e suas componentes de velocidade (linear e angular) são transformadas para o referencial SCL. Considerando-se disponíveis as velocidades do veículo relativas ao referencial SCA, conforme a EQ 4.2, e a velocidade do vento em relação a este mesmo referencial, conforme a equação

$$x_W{}^T = \left[u_W, v_W, w_W, p_W, q_W, r_W\right], \qquad \text{(EQ 4.7)}$$

a velocidade do veículo relativa ao ar é dada por

$$x_a = x - x_W . \qquad \text{(EQ 4.8)}$$

Uma definição importante para se calcular as forças aerodinâmicas, como na EQ 4.4, é a velocidade linear total relativa ao ar, dada pela equação

$$v_{ar\_tot} = \sqrt{{u_a}^2 + {v_a}^2 + {w_a}^2} \ . \qquad \text{(EQ 4.9)}$$

4.3.7 TERMOS COMPLEMENTARES

O vetor de forças dinâmicas, F_d, contém os termos de Coriolis e centrífugos do modelo dinâmico. Na presente dissertação, os termos de Coriolis são desprezados. Entretanto, conforme realizado por (RAMOS, 2002), foi introduzido o efeito do vento. Usualmente, o vento é definido no referencial inercial e suas componentes de velocidade (linear e angular) são transformadas para o referencial do dirigível para calcular, principalmente, as forças aerodinâmicas, que dependem da velocidade relativa do ar incidente no dirigível.

4.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Para o restante do trabalho considera-se, portanto, que se dispõe de um modelo matemático de dirigível qualitativamente representativo da dinâmica do veículo, mas ainda quantitativamente impreciso. O refinamento e ajuste do modelo e dos coeficientes adimensionais constituem tema para trabalhos futuros, posteriores a esta dissertação, com a possibilidade de utilizar modelos reais para determinar os coeficientes necessários e para determinar quais constantes podem se tornar funções dependentes de alguma outra variável.

Outra consideração importante diz respeito à forma de utilização das fórmulas com a OpenGL. Tendo em vista que a OpenGL trabalha com vetores separados para a posição e a orientação, o vetor 6x1 é decomposto em dois vetores, separando a parte angular da linear. O sistema de coordenadas considerado difere em alguns aspectos do sistema considerado em (RAMOS, 2002). Algumas outras questões sobre o modelo matemático adaptado são consideradas no Capítulo 5, que trata dos problemas relacionados à implementação desse modelo, bem como da implementação do mecanismo de renderização das imagens tridimensionais apresentadas pelo simulador.

5 O SIMULADOR

A simulação computacional com base em modelos matemáticos está hoje presente em todas as ciências, pois se trata de um meio de confrontar teorias com experimentação, de antecipar resultados experimentais ou de realizar experiências de outro modo inacessíveis. Os ambientes de simulação podem ser estabelecidos, dentre outras formas, a partir de modelos matemáticos de sistemas robóticos, como o que é apresentado no Capítulo 4.

Dentre as vantagens da simulação, destaca-se a possibilidade de agregar ferramentas matemáticas computacionais de análise e síntese para a realização e a avaliação de experimentos. No caso particular da simulação de veículos robóticos, é possível deixar o usuário mais familiarizado com o comportamento dinâmico do veículo. É possível ainda testar o veículo em diferentes cenários, nem sempre fáceis de serem reproduzidos experimentalmente. Métodos de controle e navegação também podem ser desenvolvidos, testados e validados antes de sua implementação no veículo real, levando-se em consideração, inclusive, aspectos como tempo e esforço computacional requeridos para a implementação real e nível de confiança que se pode depositar em um dado sistema de controle, associado à sua robustez.

Podendo ser considerada uma das principais contribuições deste trabalho, o simulador apresentado neste capítulo tem por objetivo fornecer subsídios para apoiar o desenvolvimento de algoritmos necessários para a atuação autônoma de uma equipe de dirigíveis. Sensores como câmeras, sonares, altímetros, giroscópios, dentre outros, são simulados, o que permite o desenvolvimento de algoritmos para diferentes propósitos e que utilizem diferentes recursos. O alto grau de correspondência entre as imagens renderizadas pelo simulador e as imagens reais é também uma característica marcante do simulador, permitindo o desenvolvimento de algoritmos baseados em análise de imagens.

O desenvolvimento da autonomia é um processo seqüencial e dependente do tipo de tarefa que se deseja realizar de forma autônoma. Proporcionar ao dirigível a capacidade de realizar uma tarefa de forma autônoma depende, na maioria das vezes, da realização autônoma de outras subtarefas. A navegação autônoma, por exemplo, depende da realização de várias subtarefas, que vai desde o reconhecimento de obstáculos até uma possível reconstrução tridimensional do ambiente, quando não se tem um mapa desse ambiente *a*

priori. Os algoritmos que resolvem cada uma dessas etapas podem ser desenvolvidos e incorporados ao simulador, facilitando a realização de testes e poupando recursos financeiros que seriam gastos em simulações reais. O restante do capítulo apresenta os pontos principais do desenvolvimento do simulador, as dificuldades encontradas, bem como as possibilidades de melhoria e extensão.

5.1 A IMPLEMENTAÇÃO DO MODELO MATEMÁTICO

A implementação do modelo matemático do dirigível no simulador aumenta sensivelmente a realidade da simulação, pois permite ao dirigível virtual imitar o comportamento do dirigível real, obedecendo às leis newtonianas. Com isso, a migração dos programas implementados no simulador para plataformas embarcadas reais pode ser realizada de forma menos traumática, consistindo, basicamente, na realização de ajustes nos coeficientes adimensionais do tipo de dirigível que está sendo simulado.

A implementação de um modelo matemático para um simulador de veículos robóticos aéreos consiste basicamente em resolver os problemas de cinemática e dinâmica de corpos rígidos. Como ocorre em problemas de simulação de cinemática e dinâmica de partículas, isso demanda o desenvolvimento de um conjunto de algoritmos capazes de calcular as forças atuantes no dirigível e determinar como elas alteram a posição do dirigível relativa ao ambiente em que está inserido. Entretanto, como o dirigível se trata de um corpo rígido – que se diferencia de uma partícula por ter dimensões consideráveis – tão importante quanto a posição em relação a um referencial é a orientação do dirigível com relação a um referencial. Isso significa que todos os momentos (ou torques) atuantes no dirigível, em um determinado instante do tempo, também devem ser determinados.

Os conceitos fundamentais utilizados são, principalmente, as três leis de Newton, a dinâmica básica de corpos rígidos (força e torque), operações com vetores e transformações geométricas, derivação e integração numérica. Assume-se que o leitor tenha um conhecimento básico desses assuntos. Os conceitos mais importantes estão resumidamente descritos a seguir, com algumas observações sobre como implementá-los.

5.1.1 AS VARIÁVEIS DE ESTADO

O estado de uma partícula em um tempo t consiste na posição e na velocidade da partícula. Generalizando esse conceito, pode-se definir um vetor de estados $Y(t)$ para uma simples partícula como

$$Y(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}, \tag{EQ 5.1}$$

sendo que x é dado em metros e v é dado em metros/segundo. Como se trata de um ambiente tridimensional (ou espacial), x e v são grandezas vetoriais de três elementos (ou seja, em $\Re^3$), onde cada elemento representa, respectivamente, a decomposição da grandeza representada pelo vetor nos eixos X, Y e Z do sistema de coordenadas considerado.

Para um sistema com n partículas, aumenta-se $Y(t)$ de forma que ele fique como

$$Y(t) = \begin{pmatrix} x_1(t) \\ v_1(t) \\ . \\ . \\ . \\ x_n(t) \\ v_n(t) \end{pmatrix}, \tag{EQ 5.2}$$

onde $x_i(t)$ e $v_i(t)$ são a posição e a velocidade da i-ésima partícula, respectivamente.

Como um dirigível é formado por diferentes componentes (envelope, gôndola, propulsores, profundor, leme etc.), para determinar algumas das forças e torques atuantes no dirigível é necessário saber o estado individual (ou local) de cada componente. Por exemplo, para determinar o torque exercido pela força aerodinâmica atuante no leme, é necessário saber a velocidade local do ar passando pelo leme. Contudo, nem sempre essa velocidade será igual à velocidade do ar relativa ao dirigível como um todo, tendo em vista que a velocidade de rotação (ou angular) do dirigível pode causar uma variação na velocidade do ar no componente quando o componente estiver posicionado fora do centro de gravidade. Nesse sentido, o componente pode ser caracterizado por uma partícula, e o dirigível passa a ser considerado um sistema de n partículas, cada uma representando um dos componentes.

Diferente do que ocorre com uma partícula, o estado de um corpo rígido deve conter informações sobre sua orientação e velocidade angular.

Seja $q(t)$ um quaternião (VICCI, 2001) que representa a orientação de um dirigível no instante t. A utilização de quaternião é uma forma alternativa à matriz de rotação para armazenar a orientação de um corpo no espaço, e apresenta algumas vantagens relacionadas à implementação. Uma das principais vantagens diz respeito à redução de erros numéricos acumulados provocados por sucessivas rotações bem pequenas, freqüentemente ocorridos quando se utilizam matrizes de rotação (BARAFF, 1997). Um quaternião é composto por duas partes, sendo uma escalar e outra vetorial em $\Re^3$. A representação por um quaternião q de uma rotação de θ graus sobre um vetor u é obtida pela equação

$$q = (\cos(\theta/2), sen(\theta/2) \cdot u), \quad \text{(EQ 5.3)}$$

onde a parte à esquerda da vírgula é a parte escalar e a parte à direita é a vetorial. Maiores detalhes sobre operações com quaterniões podem ser obtidos em (VICCI, 2001).

Para representar a velocidade angular no instante t, utiliza-se $w(t)$, sendo w um vetor em $\Re^3$ em que cada elemento w_x, w_y e w_z representa a velocidade de rotação, em radianos por segundo, em cada um dos eixos X, Y e Z do sistema de coordenadas considerado. Acrescentando-se $q(t)$ e $w(t)$ para determinar, respectivamente, a orientação e a velocidade angular, poder-se-ia definir $Y(t)$ para um corpo rígido como

$$Y(t) = \begin{pmatrix} x(t) \\ q(t) \\ v(t) \\ w(t) \end{pmatrix}. \quad \text{(EQ 5.4)}$$

Contudo, uma forma melhor, do ponto de vista de implementação, para armazenar a velocidade linear e a velocidade angular de um corpo rígido é através da utilização de *momentum*, por ter relação direta com a derivada das forças e dos torques que atuam no corpo. O *momentum* linear L de um corpo de massa total m é definido como

$$L = m \cdot v, \quad \text{(EQ 5.5)}$$

onde v é a velocidade do corpo.

De forma análoga, o *momentum* angular A de um corpo é definido como

$$A = I \cdot w, \tag{EQ 5.6}$$

onde I é o tensor de inércia do corpo e w é a velocidade angular. Assim como a massa m de um corpo é a resistência que o corpo tem a movimentos lineares, o tensor de inércia I é a resistência do corpo a movimentos angulares. Maiores informações sobre tensores de inércias (ou matriz de massa) podem ser obtidas em (HIBELER, 2005).

Considerando o *momentum* angular e o *momentum* linear como as novas variáveis de estado, $Y(t)$ pode agora ser definido como

$$Y(t) = \begin{pmatrix} x(t) \\ q(t) \\ L(t) \\ A(t) \end{pmatrix}. \tag{EQ 5.7}$$

Definidas as variáveis de estado necessárias para representar as informações espaciais (a posição e a orientação) e as informações das velocidades (*momentum* linear e *momentum* angular), o próximo passo consiste em determinar como essas grandezas irão variar com o tempo, passo esse necessário para simular a movimentação do dirigível. A subseção 5.1.2 trata especificamente desse assunto.

5.1.2 A INTEGRAÇÃO DOS PASSOS DE SIMULAÇÃO

Como já apresentado em subseções anteriores, um dirigível navegando em um ambiente aberto sofre a ação de diversas forças externas, como a gravidade, o vento, a propulsão dos motores, o arrasto, entre outras. São essas as forças que irão provocar a alteração dos valores das variáveis de estado do dirigível virtual ao longo do tempo. A massa m do dirigível e o seu tensor de inércia I são constantes, pois, para o projeto em questão, considera-se que o dirigível usa baterias como fonte de energia e não, combustíveis. Caso seja adotado o uso de combustíveis, deve-se considerar que a queima de combustível acarreta uma perda considerável de massa, logo, m e I não seriam constantes e deveriam passar a ser consideradas variáveis de estado.

Armazenando-se apenas o *momentum* linear $L(t)$ e o *momentum* angular $A(t)$, em qualquer instante de tempo t pode-se computar a velocidade linear $v(t)$ e $w(t)$ por

$$v(t) = \frac{L(t)}{m} \quad \text{e} \quad w(t) = I(t)^{-1} \cdot A(t). \tag{EQ 5.8}$$

Para determinar o *momentum* linear e o *momentum* angular do dirigível é necessário conhecer todas as forças e torques que agem sobre ele. O cômputo das forças e torques é tratado com exclusividade na subseção 5.2.4. Seja $F(t)$ a soma de todas as forças e $\tau(t)$ a soma de todos os torques que agem no dirigível no instante de tempo t. Pode-se determinar $L(t)$ e $A(t)$ através de

$$L(t) = F(t) \cdot dt \quad \text{e} \quad A(t) = \tau(t) \cdot dt\,, \tag{EQ 5.9}$$

onde dt é o tamanho do passo de simulação em segundos. Entretanto, para computar corretamente a posição e a orientação do dirigível no instante t, é necessário ter o *momentum* linear e o *momentum* angular acumulados ao longo da simulação, o que corresponde a realizar a seguinte operação a cada passo da simulação:

$$L = L + F(t) \cdot dt \quad \text{e} \quad A = A + \tau(t) \cdot dt\,, \tag{EQ 5.10}$$

onde L e A são, respectivamente, o *momentum* linear e o *momentum* angular acumulados. Agora, pode-se definir

$$\frac{d}{dt}Y(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ q(t) \\ L(t) \\ A(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ [w(t)q(t)]/2 \\ F(t) \\ \tau(t) \end{pmatrix}. \tag{EQ 5.11}$$

A multiplicação de $w(t)$ por $q(t)$ consiste na multiplicação do quaternião $q_w = (0, w(t))$ por $q(t)$. A partir das formulações supracitadas, é apresentado a seguir um algoritmo em alto nível que demonstra a integração usando o método de Euler (KRUS, 2007).

No algoritmo em pseudocódigo da FIG. 5.1, a linha 01 recebe como parâmetro o valor de dt, que consiste no tamanho do passo de simulação (ou integração) em segundos. Na linha 03, há uma chamada para um método que calcula todas as forças e torques atuantes no dirigível no tempo t. Detalhes de implementação desse método são apresentados na subseção 5.2.4. Nas linhas 04 e 05, são calculados respectivamente o *momentum* linear e o *momentum* angular acumulados, usando a EQ 5.10. Nas linhas 06 e 07, usam-se os valores obtidos nas linhas 04 e 05 para calcular, respectivamente, a velocidade linear e a velocidade angular, de acordo com a EQ 5.8. Nas linhas 08 e 09, a posição e a orientação são determinadas, de acordo com a EQ 5.11. Tendo a nova posição e orientação do corpo rígido, os objetos tridimensionais podem

ser atualizados para que suas posições e orientações fiquem consistentes com os novos valores das variáveis de estado.

```
01. procedimento StepSimulationEuler(dt);
02. inicio
03.   ComputeForcesAndTorques();
04.   L ← L + [F(t) * dt];
05.   A ← A + [τ(t) * dt];
06.   v ← L / m;
07.   w ← A * I^-1;
08.   x ← x + (v * dt);
09.   q ← q + [(w * q) * 0.5 * dt];
10. fim;
```

FIG. 5.1 – Algoritmo de execução do passo de simulação usando o método de Euler.

5.1.3 AS CLASSES IMPLEMENTADAS

Visando facilitar a implementação do modelo matemático do dirigível no simulador, bem como sua manutenção e extensão, foi utilizado o paradigma da orientação a objetos. Um diagrama de classes da UML com as classes implementadas é apresentado na FIG. 5.2. A classe principal é a classe *RigidBody*. Essa classe representa um corpo rígido genérico. A classe implementa métodos e contém atributos que são comuns aos corpos rígidos em geral. Se for necessário representar um veículo com características e comportamentos específicos, a classe pode ser estendida. A classe *Blimp* é um exemplo de uma classe derivada de *RigidBody*, herdando suas características e comportamentos. Na classe *RigidBody*, o atributo *mass* é um escalar e armazena a massa total do corpo rígido. O atributo *centerOfGravity* armazena a posição do centro de gravidade calculada durante a inicialização de um objeto do tipo *RigidBody*. Quando o corpo rígido é criado, sua posição é determinada de acordo com o Sistema de Coordenadas do Ambiente (SCA), que consiste em um sistema de coordenadas com a origem fixada em um referencial do ambiente em que se encontra o corpo, com o eixo Z relacionado à altitude, X na direção Norte e Y na direção Leste.

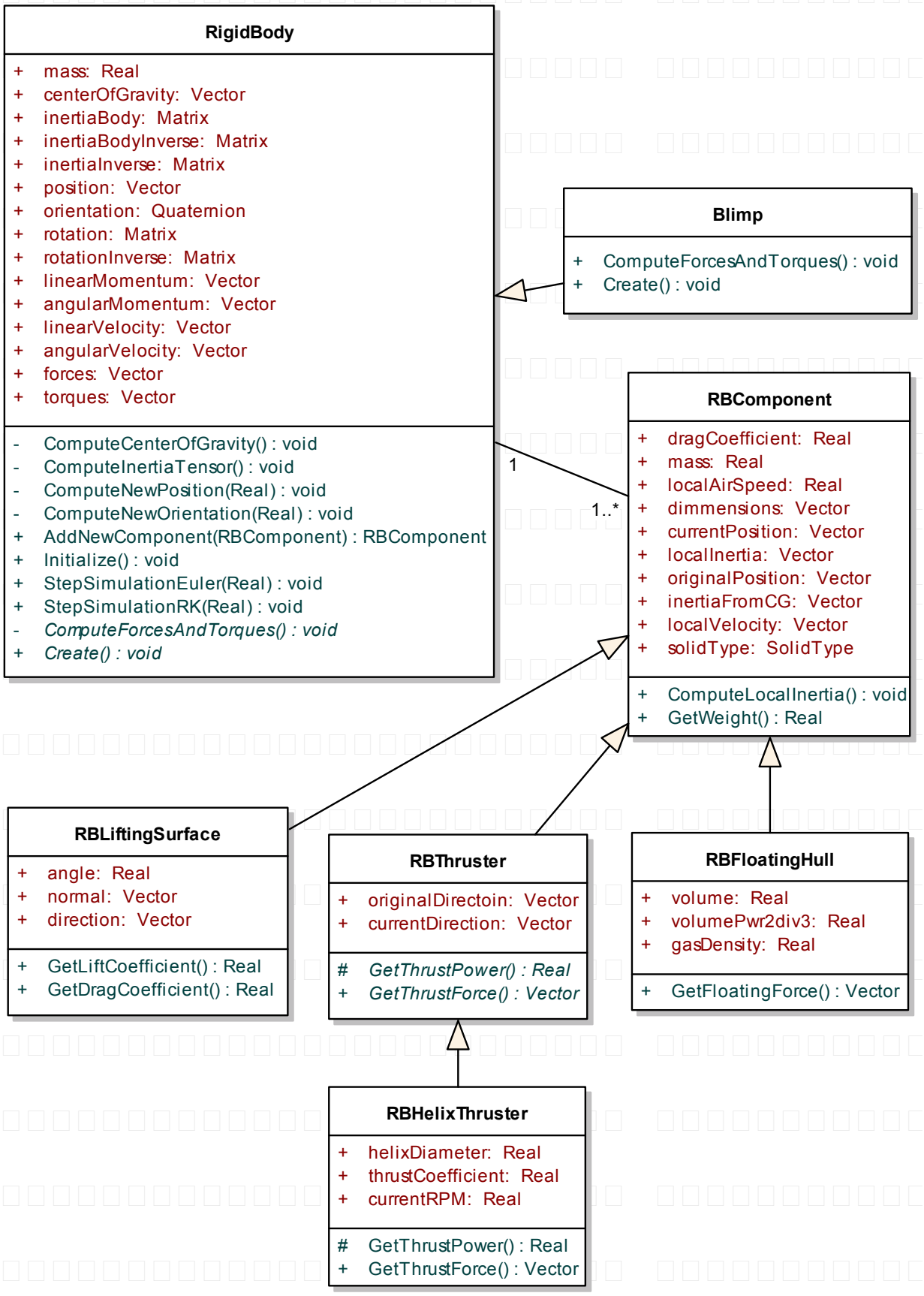

FIG. 5.2 – Diagrama de classes do modelo matemático do dirigível.

Sempre, pelo menos um objeto *RBComponent* ou derivado deste deve ser adicionado ao corpo rígido, sendo que cada componente adicionado possui uma posição relativa ao centro de gravidade do corpo rígido, ou seja, com relação a um Sistema de Coordenadas Local (SCL), que tem a origem no centro de gravidade do corpo rígido. Quando necessário, é possível computar a posição de cada componente relativa ao SCA através do quaternião de orientação *orientation*. Para isso, é necessário rotacionar o vetor posição do componente (*currentPosition*) de acordo com a orientação em *orientation*, convertendo o quaternião para uma matriz de rotação (armazenada no atributo *rotation*). Feito isso, deve-se multiplicar a matriz pelo vetor posição e somar o resultado à posição do corpo rígido em si, como na equação

$$x_i(t)_{SCA} = R(t) * x_i(t)_{SCL} + x(t), \qquad \text{(EQ 5.12)}$$

onde $x_i(t)_{SCA}$ é a posição do *i*-ésimo componente relativa ao SCA, $R(t)$ é a matriz de rotação computada a partir do quaternião *orientation*, $x_i(t)_{SCL}$ é a posição do *i*-ésimo componente relativa ao SCL e $x(t)$ é a posição do corpo rígido, que sempre é relativa ao SCA. Todos esses valores são considerados para um instante *t*.

Em algumas situações, a velocidade local de um componente do corpo rígido no SCA é importante, principalmente quando o componente possui características aerodinâmicas que geram forças e torques no corpo rígido. Nesse caso, a velocidade do vento no componente deve ser determinada e, para isso, deve-se calcular a velocidade do componente relativa ao SCA, utilizando a equação

$$\dot{x}_i(t)_{SCA} = w(t) \times (x_i(t)_{SCA} - x(t)) + v(t), \qquad \text{(EQ 5.13)}$$

onde $\dot{x}_i(t)_{SCA}$ é a velocidade do *i*-ésimo componente em relação ao SCA, $w(t)$ é a velocidade angular do corpo rígido, $x_i(t)_{SCA}$ é a posição do *i*-ésimo componente relativa ao SCA, $x(t)$ é a posição do corpo rígido, $v(t)$ é a velocidade do corpo rígido no SCA e o operador $\times$ é o produto vetorial, todos considerados para o instante de tempo *t*. A classe *RBLiftingSurface* possui atributos específicos para representar componentes com características exclusivamente aerodinâmicas, como lemes e profundores.

Um requisito básico para facilitar as conversões entre coordenadas no SCA e no SCL é que a posição do centro de gravidade do corpo rígido no SCL esteja no ponto (0, 0, 0), para os

eixos X, Y e Z, respectivamente. Devido a esse requisito, durante a inicialização de um objeto do tipo *RigidBody*, o centro de gravidade é recalculado, de forma que passe a ficar no ponto (0, 0, 0) no SCL, o que incorre em um reajuste da posição de todos componentes do corpo rígido.

A coexistência de ambos referenciais no processo de simulação introduz a necessidade de transformações do SCL para o SCA e vice-versa, o que leva a algumas particularidades no algoritmo de simulação e no processo de integração numérica, bem como no cálculo das forças e torques. A subseção seguinte complementa a justificação da criação de diferentes classes para a implementação do modelo matemático.

5.1.4 A DETERMINAÇÃO DAS FORÇAS E DOS TORQUES

As classes apresentadas no diagrama da FIG. 5.2 são suficientes para a simulação de dirigíveis, que são constituídos basicamente de quatro tipos de componentes: os componentes comuns (gôndola, sensores etc.), que podem ser representados pela classe *RBComponent*, por não interferirem deliberadamente na movimentação do dirigível e por não oferecerem propulsão ou forças aerodinâmicas consideráveis; o componente flutuante (envelope), representado por *RBFloatingHull*, responsável pela força aerostática que mantém o dirigível suspenso no ar; os componentes propulsores que usam hélice, representados pela classe *RBHelixThruster*; e os componentes aerodinâmicos, representados pela classe *RBLiftingSurface*.

Essas classes podem ser estendidas para representarem outros tipos de veículos, como aviões, planadores, helicópteros, carros, dentre outros. A FIG. 5.3 mostra um dirigível e seus componentes, nomeados com a classe que os representa. A classe *RBComponent* detém os atributos comuns aos tipos de componentes usados por corpos rígidos. O atributo *mass* representa a massa de um componente. A massa total do corpo rígido é obtida através da soma das massas de cada um dos componentes que o constituem. O atributo *dragCoefficient*, *dimmensions* e *localAirSpeed* são usados para determinar a força de arrasto dinâmico que atua no componente, de acordo a EQ 5.14,

$$F_i = [0.5 \cdot \rho \cdot s_i^{\,2} \cdot A_i] \cdot (-v_i), \qquad \text{(EQ 5.14)}$$

onde F_i é a força de arrasto que atua no componente, ρ é a densidade do ar, s_i é um escalar que representa a velocidade do ar incidente no componente, A_i é a área de incidência no componente e v_i é o vetor velocidade do componente no SCA.

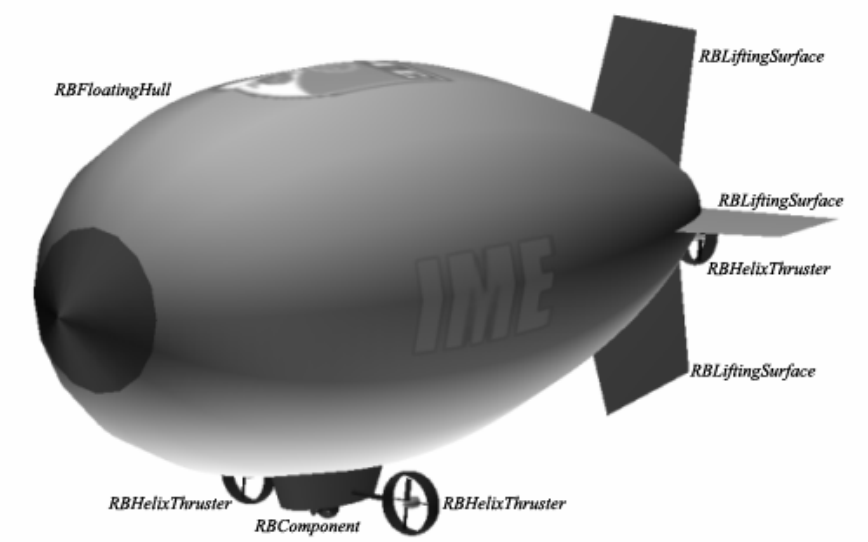

FIG. 5.3 – Um dirigível e seus componentes.

O termo final v_i da EQ 5.14 ocorre devido à força de arrasto atuar na mesma direção e em sentido contrário à velocidade do componente no SCA. Assim como em equações anteriores, os termos com índice *i* correspondem a grandezas pertinentes ao *i*-ésimo componente do corpo rígido. Basicamente, o peso e a força de arrasto são as únicas forças relevantes que atuam nos componentes da classe *RBComponent*.

A classe *RBFloatingHull* é derivada de *RBComponent* e, por isso, herda suas características. O diferencial de um componente do tipo *RBFloatingHull* é que ele exerce uma força aerostática no corpo rígido, de mesma direção e com sentido oposto à força da gravidade. Essa classe representa o envelope do dirigível. A força exercida é calculada de acordo com o volume e com a densidade do gás contido no envelope, de acordo com EQ 5.15,

$$F_i = (\rho_{ar} - \rho_{gás}) \cdot V_i, \qquad \text{(EQ 5.15)}$$

onde F_i é a força de flutuação, ρ_{ar} é a densidade do ar, $\rho_{gás}$ é a densidade do gás dentro do envelope (usualmente hélio ou hidrogênio) e V_i é o volume do envelope (KHOURY e GILLETT, 1999).

A classe *RBLiftingSurface*, assim como *RBFloatingHull*, é derivada de *RBComponent*. Componentes da classe *RBLiftingSurface* exercem sua força de acordo com a velocidade e o ângulo de incidência do ar passando por eles. A força tem a direção do vetor normal à superfície do componente e pode provocar arrasto aerodinâmico, sustentação aerodinâmica e, conseqüentemente, um torque.

Os componentes de propulsão do tipo de dirigível considerado neste trabalho utilizam hélices e são representados pela classe *RBHelixThruster*. A força exercida pelo propulsor obedece à EQ 5.16, definida como

$$F_i = (c_i \cdot \rho \cdot n_i^{\,2} \cdot D_i^{\,4}) \cdot d_i\,, \qquad \text{(EQ 5.16)}$$

onde F_i é a força exercida pelo propulsor, c_i é o coeficiente de propulsão da hélice, ρ é a densidade do ar, D é o diâmetro da hélice e d_i é o vetor de direção do propulsor no SCL (RAMOS, 2002).

A força exercida pelo peso de cada um dos componentes age diretamente no centro de gravidade (ou centro de massa) do dirigível, e é calculada na sua inicialização. Essa força corresponde à soma das massas dos componentes multiplicada pelo vetor gravidade (0, 0, g), sendo que, neste caso, g é constante, mas pode facilmente ser uma função calculada de acordo com a altitude do dirigível e a massa do planeta Terra. A força da gravidade sempre atua no sentido negativo do eixo Z do SCA.

O torque exercido no dirigível, responsável pela alteração de sua orientação, é calculado de acordo com todas as forças que atuam individualmente nos componentes posicionados fora do centro de gravidade do dirigível. Isso leva a concluir que não há torque exercido pela força da gravidade, devido ao próprio conceito de centro de gravidade. Para calcular o torque exercido pela força atuante em um componente, é utilizada a EQ 5.17, definida como

$$\tau_i = F_i \times x_i\,, \qquad \text{(EQ 5.17)}$$

onde τ_i é o torque exercido pelo i-ésimo componente, F_i é o vetor força que atua no componente, x_i é o vetor posição do componente no SCL e o operador $\times$ corresponde ao produto vetorial (HIBELER, 2005).

Como cada um dos componentes exerce individualmente força e torque no dirigível. A força total F_t e o torque total τ_t são obtidos através da EQ 5.18, definida como

$$\tau_t = \sum_{i=1}^{n} \tau_i \quad \text{e} \quad F_t = \left(\sum_{i=1}^{n} F_i \right) + F_g , \qquad \text{(EQ 5.18)}$$

onde n é a quantidade de componentes que compõe o dirigível, τ_i é o torque exercido pelo i-ésimo componente, F_i é a força que atua no i-ésimo componente e F_g é a força da gravidade, atuante no centro de gravidade do dirigível.

Observando as equações anteriores relacionadas ao cálculo das forças, pode-se notar que o vetor F_t estará no SCL. Devido a isso, para calcular a nova posição do dirigível no SCA, é preciso converter F_t para o SCA. Com a força e o torque totais calculados, o algoritmo de integração pode prosseguir com os cálculos dos novos valores das variáveis de estado do dirigível.

5.1.5 O TENSOR DE INÉRCIA

Cada componente que constitui um corpo rígido possui massa, e, por ter dimensões consideráveis, possui um vetor de inércia. Assim, o centro de gravidade do veículo, bem como sua massa e seu tensor de inércia, são calculados de acordo com informações de cada um dos componentes.

A partir da combinação do vetor de inércia de cada um dos componentes – que para sólidos regulares pode ser obtido através de fórmulas clássicas facilmente encontradas na literatura relacionada – estabelecem-se os termos da diagonal principal do tensor de inércia do corpo rígido. Desses termos diagonais, calculam-se os produtos de inércia para determinar os termos não-diagonais da matriz do tensor de inércia, necessários para complementar o tensor de inércia e prepará-lo para causar resistência a movimentos angulares em qualquer eixo que passe pelo centro de gravidade.

Em termos gerais, o tensor de inércia consiste em um fator escalar entre o *momentum* angular $A(t)$ e a velocidade angular $w(t)$. Cada componente do dirigível exerce uma influência no tensor de inércia proporcional ao quadrado da distância do componente considerado ao centro de gravidade do dirigível. Como, na maioria dos casos, os dirigíveis são simétricos com relação ao plano formado pelos eixos XZ do SCL, a resistência a movimentos angulares

no eixo X é igual para qualquer sentido da rotação. Foge ao escopo desse trabalho esgotar o assunto sobre tensores de inércia, que, por si só, é assunto para um capítulo inteiro. Um estudo mais aprofundado pode ser encontrado em (HIBELER, 2005).

5.1.6 MÉTODOS DE INTEGRAÇÃO NUMÉRICA

Existem vários métodos de integração utilizados para determinar os valores futuros das variáveis de estado em simuladores semelhantes ao desenvolvido neste trabalho (KRUS, 2007). Dentre eles, por questões práticas, foi escolhido o método de Euler. O método Runge-Kutta, por ser bastante utilizado, também é apresentado, apesar de não estar implementado na versão atual do simulador. O primeiro método tem a característica de ser mais rápido, porém menos preciso que o segundo. O segundo, apesar de mais preciso, requer maior esforço computacional para ser executado, podendo comprometer o desempenho da simulação.

Para simulações em que as variáveis de estado sofrem alterações consideráveis em um pequeno espaço de tempo, o método Runge-Kutta é mais aconselhado. Para simulações em que as variáveis de estado se alteram de forma mais suave, o método de Euler pode ser utilizado, pois o erro acumulado é bem tolerado. Em um simulador de aviões, por exemplo, em que a velocidade pode ultrapassar os 800 quilômetros horários, em um intervalo de tempo de 10 milissegundos o deslocamento é superior a 2 metros. Nessa situação, a escolha do método de integração utilizado pode interferir gravemente no resultado da simulação. Um erro de 2 metros pode ser crucial para determinar se o avião vai ou não colidir com outro durante um vôo simulado, podendo causar uma divergência entre a simulação e a realidade.

No caso de uma simulação de dirigíveis de pequeno porte, a velocidade desses raramente ultrapassa os 60 quilômetros horários. Em um intervalo de tempo de 10 milissegundos, o deslocamento é inferior a 17 centímetros. Adotando uma margem de segurança mínima para a distância entre o dirigível e outros objetos da simulação (chão, obstáculos etc.), o erro provocado pelo método de integração é bem tolerado.

Para tratar o método de Euler e outros métodos de integração numérica com mais rigor matemático, é preciso compreender o *Teorema das Séries de Taylor* (BOURG, 2002). O teorema de Taylor permite aproximar o valor de uma função em algum ponto sabendo alguma coisa sobre a função e sua derivada em algum outro ponto. Essa aproximação é expressa pela série polinomial infinita, que é da forma

$$y(x+\Delta x) = y(x) + (\Delta x)\cdot \dot{y}(x) + [(\Delta x)^2 / 2!]\cdot \ddot{y}(x) + [(\Delta x)^3 / 3!]\cdot \dddot{y}(x) + \ldots, \quad \text{(EQ 5.19)}$$

onde y é uma função de x e $(x + \Delta x)$ é o novo valor de x para o qual se deseja aproximar y.

No caso de uma equação de movimento, uma função passível de aproximação é a velocidade como função do tempo. Escrevendo $v(t)$ ao invés de $y(t)$ na EQ 5.19 tem-se

$$v(t+\Delta t) = v(t) + (\Delta t)\cdot \dot{v}(t) + [(\Delta t)^2 / 2!]\cdot \ddot{v}(t) + [(\Delta t)^3 / 3!]\cdot \dddot{v}(t) + \ldots. \quad \text{(EQ 5.20)}$$

O que se deseja é encontrar v no tempo $t + \Delta t$, dado v no tempo t e sua derivada no tempo t. Como uma primeira aproximação, como não se sabe nada a respeito da segunda derivada de v em diante, pode-se truncar a série polinomial, resultando em

$$v(t+\Delta t) = v(t) + (\Delta t)\cdot \dot{v}(t). \quad \text{(EQ 5.21)}$$

Essa é a fórmula de integração de Euler. Uma vez que a fórmula considera apenas os termos até a primeira derivada de v, o restante é o erro de truncamento. Essa aproximação é plausível porque quanto mais adiante se prossegue na série de Taylor, menor é o valor do último termo e menor é a influência que ele exerce no valor da aproximação. Uma vez que Δt é um número bem pequeno, Δt^2 é bem menor, assim como Δt^3, Δt^4 etc. Ainda, como esses valores aparecem no numerador (Δt^2 / 2! etc.), cada termo sucessivo de maior ordem será sucessivamente menor, pois será também divido por algum valor. Esse método possui um erro da ordem de (Δt^2).

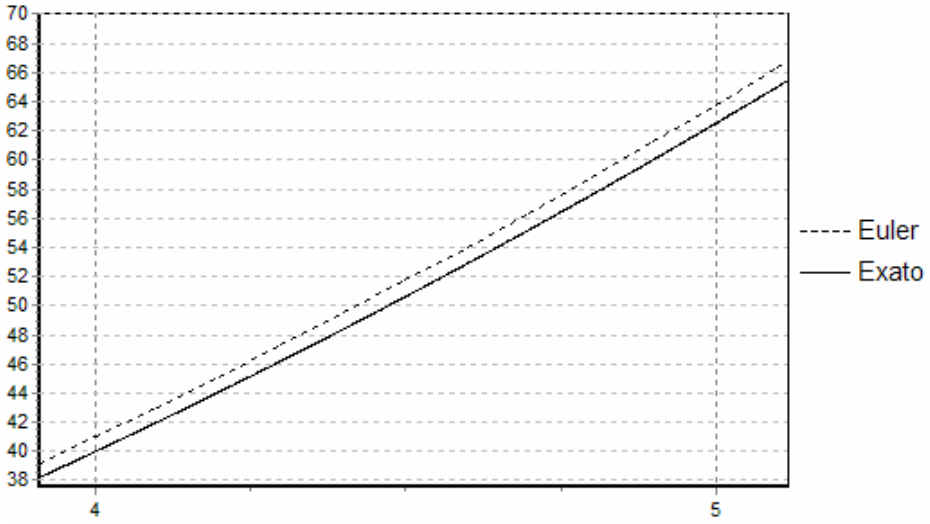

FIG. 5.4 – Comparação do método de Euler com a solução exata.

A FIG. 5.4 mostra um gráfico comparativo do cálculo da posição de um corpo de massa igual a 2 kg em função do tempo, partindo do repouso, sofrendo a aplicação de uma força constante de 10 N. O eixo horizontal mostra o intervalo de tempo em segundos e o eixo vertical mostra o deslocamento em metros. Nesse caso, usando um passo de simulação de 100 milissegundos, o erro médio para a variação da posição durante os 10 primeiros segundos foi de 1,68%. Usando um passo de simulação de 10 milissegundos, o erro cai para menos de 0,1%, um erro tolerável para o tipo de simulação em questão. Outros métodos de integração podem ser usados para simulações em tempo real, e a maioria deles procura encontrar um ponto de equilíbrio entre precisão e desempenho.

Uma forma de reduzir o erro de truncamento e, conseqüentemente, aumentar a precisão da aproximação, pode ser obtida se forem considerados mais termos da série de Taylor. Para isso, os termos derivados de segunda ordem em diante devem ser determinados, e é onde se encontra a maior dificuldade. Em (BOURG, 2002), há também um método de Euler "aprimorado" que realiza essa tarefa, considerando apenas mais um termo. Há ainda, nessa mesma linha, o método Runge-Kutta de quarta ordem, que faz uma aproximação ainda melhor, reduzindo o erro de truncamento para a ordem de (Δt^5). O método Runge-Kutta é amplamente utilizado em simulações em tempo real por possuir um bom desempenho, mesmo considerando mais termos do que o método de Euler. Porém, sua implementação é mais complexa e, mesmo tendo bom desempenho, requer cerca de 4 vezes mais computação do que o método de Euler.

5.1.7 OBSERVAÇÕES SOBRE A IMPLEMENTAÇÃO DO MODELO MATEMÁTICO

A implementação do modelo matemático de um corpo rígido, em particular, de um dirigível, consiste basicamente em definir o comportamento das variáveis de estado a partir de um conjunto de entradas. Não faz sentido desenvolver um simulador para um trabalho científico como o presente trabalho sem a implementação de tal modelo. No entanto, a implementação do modelo matemático de um corpo rígido requer o desenvolvimento de uma série de funções extras para lidar com alguns tipos de grandezas.

As operações necessárias à manipulação das grandezas relacionadas ao movimento de um dirigível são basicamente operações sobre vetores, matrizes e quaterniões, tendo em vista que essas grandezas estão no espaço tridimensional de atuação do dirigível. Isso demandou a implementação de uma biblioteca com mais de quarenta funções para operações e conversões

entre vetores, matrizes e quaterniões, o que não deixa de ser um complemento à contribuição deste trabalho. Operações como adição de vetores, de matrizes, multiplicação de vetores e matrizes por um escalar, produto vetorial, conversão de quaternião para matriz de rotação, entre outras, são algumas das operações disponíveis nessa biblioteca.

Ainda, apesar da biblioteca de funções ter sido codificada na linguagem *Object Pascal*, assim como o restante do simulador, a conversão para outra linguagem de programação é um trabalho bem menos árduo do que a reimplementação da biblioteca partindo apenas das fórmulas e teorias.

Outra questão que merece atenção ao se implementar um modelo matemático é a realização de testes e a busca por erros. Apesar de a ferramenta utilizada (Delphi 7.0) oferecer diversas funções de localização de erros, encontrar erros nesse tipo de programa é uma tarefa trabalhosa. A complicação ocorre porque a atualização das variáveis de estado ocorre mais de 100 vezes por segundo, impossibilitando a inspeção passo a passo do valor de cada variável durante a execução da simulação. Para resolver essa questão foi adotada uma técnica de rastreamento e observação das variáveis após a execução de uma simulação.

A cada passo da simulação, os valores das variáveis a serem inspecionadas são escritos seqüencialmente em um arquivo texto, que pode ser analisado posteriormente. Apesar de parecer rudimentar, é possível detectar anomalias nas seqüências de variação das variáveis de estado rastreadas, desde que se saiba o que esperar do comportamento do dirigível em uma determinada situação da simulação. Vale ressaltar que, tanto essa quanto as outras observações, não estão relacionadas a uma linguagem de programação específica. Se por alguma questão o simulador precisar ser convertido para outra linguagem, essas observações devem ser consideradas.

5.2 A RENDERIZAÇÃO DAS IMAGENS TRIDIMENSIONAIS

Um dos principais objetivos da Computação Gráfica é criar imagens sintéticas realistas. O realismo visual depende das técnicas de tratamento computacional aplicadas aos objetos sintéticos gerados (por modelagem de sólidos, partículas, fractais ou qualquer técnica de geração). O objetivo é chegar o mais perto possível da realidade que se teria se os objetos fossem construídos e filmados.

O realismo é um fator crucial no entretenimento, na educação e, na maioria das vezes, em simulações. Em simuladores, o realismo tem ainda mais importância quando as imagens simuladas devem passar por algum processo de análise para se extrair algum dado importante a partir da imagem. Esses dados poderiam servir, por exemplo, para localizar um alvo ou para construir um mapa tridimensional que represente o ambiente como um todo ou apenas em parte.

No caso do simulador desenvolvido nesta dissertação, as imagens são fundamentais para o desenvolvimento de algoritmos inteligentes de navegação autônoma baseada em imagens. Pensando em termos práticos, se um dirigível real possuir mais de uma câmera embarcada, é possível utilizar técnicas de visão estereoscópica para fazer uma reconstrução tridimensional do ambiente, desde que as câmeras estejam calibradas (SINOPOLI, 2001).

Outra possibilidade de uso útil das imagens está na localização de alvos. O reconhecimento de padrões em imagens é um assunto muito pesquisado e com resultados satisfatórios. O desenvolvimento de algoritmos para o reconhecimento de formas geométricas é bem comum nessa área, como é o caso dos algoritmos para reconhecimento ótico de caracteres. Utilizando-se técnicas semelhantes, é possível desenvolver algoritmos para reconhecer objetos mais complexos, como armas, alvos na superfície terrestre etc.

Quando se dispõe de um conjunto seqüencial de imagens, geradas a partir da animação de alguma cena, é possível determinar alguns pontos de interesse nas imagens e acompanhá-los. Com isso, consegue-se determinar até mesmo a velocidade com a qual os pontos estão se movimentando. Para o problema geral apresentado neste trabalho, caso se tenha um dirigível com apenas uma câmera embarcada, é possível utilizar a seqüência de imagens da câmera para determinar a velocidade e a trajetória do dirigível.

A animação das imagens geradas é um fator crucial para o realismo de uma simulação. No contexto da Ciência da Computação, há o termo Tempo Real (MOLLER e HAINES, 2002), que expressa genericamente computações realizadas num intervalo de tempo pequeno e imediato. Especificamente no contexto da Computação Gráfica, tempo-real significa que os cálculos necessários para síntese de uma imagem são rápidos o suficiente para acontecer a taxas ao redor de 15 imagens (ou *frames*) por segundo (fps), no mínimo. Em outras palavras, as imagens sintéticas estão sendo criadas com rapidez suficiente para aceitar interações de

controle ou outras, pelo usuário ou por outros sistemas, de forma que a resposta seja sempre adequada.

A utilização da biblioteca OpenGL facilita a aproximação entre o virtual e o real. A biblioteca de componentes GLScene[1] permite utilizar componentes para criar imagens tridimensionais. A seguir, são apresentadas as técnicas utilizadas na tentativa de aproximar as imagens do simulador às imagens de objetos e ambientes reais.

5.2.1 A RENDERIZAÇÃO DOS DIRIGÍVEIS

Com relação aos aspectos visuais, um dirigível pode ser decomposto em alguns componentes principais. O componente mais notável é o envelope de gás, que é a maior parte do dirigível. A forma geométrica comum mais parecida com o envelope é a forma oval, ou elipsóide. Entretanto, a maioria dos dirigíveis atuais tem a cauda mais afinada e a parte frontal mais encorpada, apresentando um formato parecido com uma gota na horizontal.

Como apresentado na subseção 3.3.6, a biblioteca OpenGL renderiza sólidos a partir de polígonos, que, por sua vez, são criados a partir de vértices. Portanto, a primeira coisa a ser feita é determinar os vértices para a formação dos polígonos. O componente *TGLRevolutionSolid* da biblioteca de componentes GLScene realiza essa tarefa recebendo os pontos de apenas uma das seqüências longitudinais de pontos.

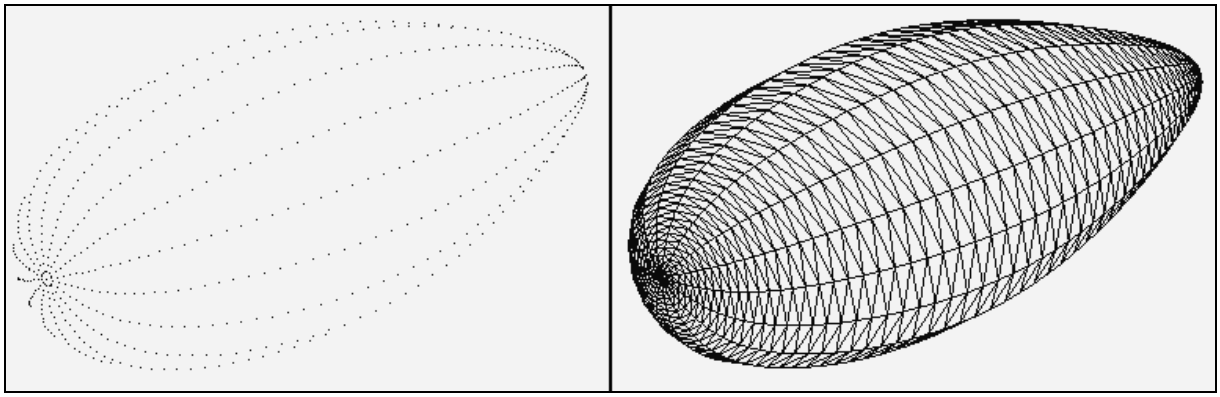

FIG. 5.5 – As duas primeiras etapas para renderizar o envelope.

[1] Disponível gratuitamente em *www.glscene.org*, sob licença de código fonte aberto.

A FIG. 5.5 ilustra os dois primeiros processos para renderizar o envelope. A parte esquerda mostra os vértices utilizados para a criação das faces triangulares, vistas na parte direita. As faces triangulares compõem o sólido. A próxima etapa consiste em aplicar uma textura às faces dos polígonos (triângulos) para que a forma adquira um aspecto mais real. A FIG. 5.6 mostra o resultado final do envelope, com a aplicação da textura e com efeitos de luz. O componente *TRevolutionSolid* também é usado para criar outros componentes do dirigível através de sólidos de revolução, como os componentes que envolvem os propulsores, os motores (miolos) dos propulsores e a gôndola. Os sólidos de revolução podem ter seu uso estendido se for utilizado um fator de escalonamento em algum dos eixos coordenados, esticando ou encolhendo o sólido em um dos eixos. A gôndola, por exemplo, é um sólido de revolução esticado longitudinalmente (eixo X).

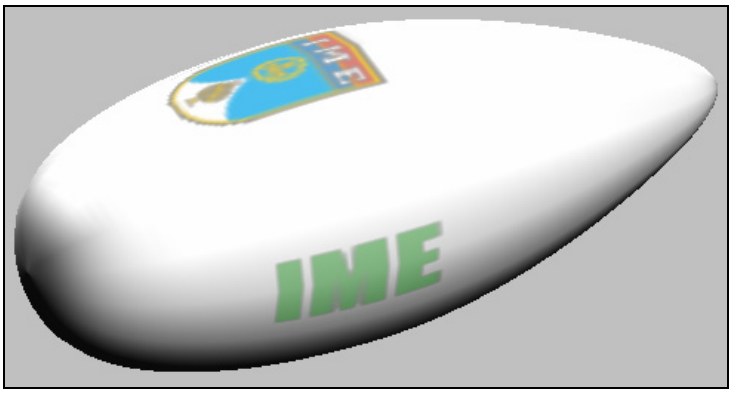

FIG. 5.6 – Resultado final da renderização do envelope.

Outras partes do dirigível, como as pás das hélices e os lemes de cauda, são formadas por planos. Um plano é representado pelo componente *TGLPlane* da biblioteca GLScene. O eixo entre os propulsores da gôndola é representado pelo componente *TGLCylinder* da GLScene, que renderiza um cilindro. A FIG. 5.7 exibe, à esquerda, a representação aramada (por triângulos) e, à direita, o resultado final da renderização de um dirigível com as suas partes principais.

Na versão atual do simulador, os únicos objetos que se movem pelo ambiente durante uma simulação são os dirigíveis. Outros objetos tridimensionais estáticos podem ser renderizados no ambiente de simulação. Esse assunto é tratado na subseção 5.2.3.

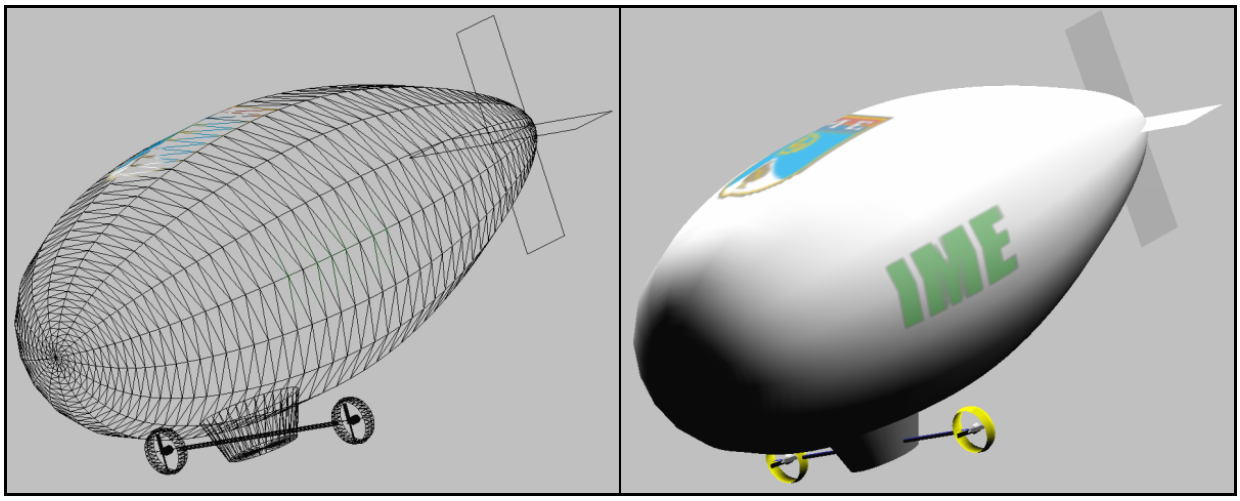

FIG. 5.7 – Representação aramada e final de um dirigível com todos os componentes.

5.2.2 A RENDERIZAÇÃO DO TERRENO

A renderização de terrenos, por si só, é um tópico de pesquisa bastante abrangente. Os aspectos mais relevantes que devem ser considerados ao se pensar em renderizar um terreno estão relacionados ao seu relevo e à sua textura.

A representação de terrenos é fundamental para simular ambientes abertos, e a representação topográfica de terrenos reais é um aspecto importante para o tipo de simulação em questão neste trabalho, pois o relevo de um terreno pode ter influência direta na navegação de um dirigível. Como exemplo, considere um terreno quadrado com uma montanha bem no centro. Um dirigível sobrevoando esse terreno poderia passar pela montanha por uma de suas laterais ou por cima dela. Um algoritmo ótimo de planejamento de trajetória faria o dirigível tomar o caminho de menor custo, em relação ao gasto de energia ou ao tempo de deslocamento.

Um método muito utilizado para renderizar terrenos é o método desenvolvido por (RÖETTGER *et al.*, 1998). A biblioteca GLScene possui um componente específico para esse fim. Trata-se do *TGLTerrainRenderer*. Esse componente foi projetado especificamente para realizar a renderização eficiente definida explicitamente ou proceduralmente através de um *heightfield* (mapa de alturas, ou mapa de relevo), que constitui em uma estrutura de dados matricial com valores de altura. Essa técnica não permite a renderização de terrenos com determinadas irregularidades, como cavernas. O algoritmo ROAM (DUCHAINEAU *et al.*,

1997) é usado para reduzir a quantidade de vértices, dependendo do ponto de vista do espectador do cenário renderizado.

Para realizar a renderização, deve-se usar uma imagem em um formato especial. Como uma imagem consiste em uma matriz de pixels, onde cada célula da matriz tem o valor da cor de cada pixel, basta considerar o valor da cor como o valor da altura. Na FIG. 5.8 (a), é mostrado um exemplo de uma imagem usada como mapa de relevo para renderizar um terreno. A imagem está em escala de cinza, com tons que podem variar de 0 a 255, sendo que, quanto maior o valor, mais claro o pixel e, quanto menor, mais escuro. Dessa forma, quanto mais escura a região na imagem, maior é a depressão naquele ponto. Na FIG. 5.8 (b), é mostrado um exemplo de textura que pode ser aplicada em um terreno. Este exemplo de textura remete a um terreno semi-desértico.

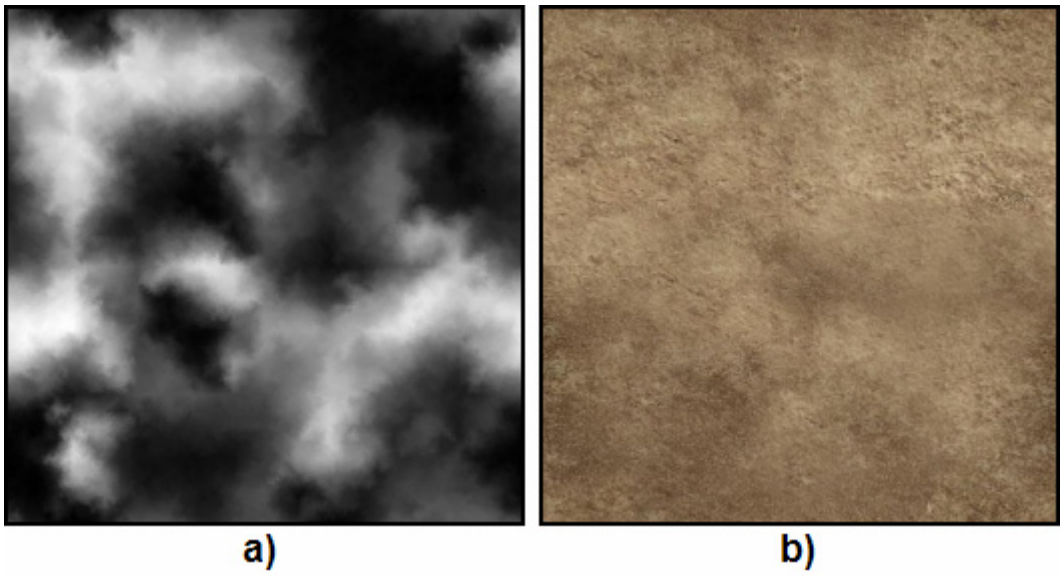

FIG. 5.8 – Exemplos de mapa de relevo (a) e de textura (b) de um terreno.

Um fator de escala pode ser utilizado nos três eixos coordenados para definir a proporção de cada célula do terreno nos eixos X, Y e Z com relação ao pixel. Considere ainda, como exemplo, o mapa de relevo da FIG. 5.8 (a), que possui 256 x 256 pixels. Seja a altura da imagem correspondente ao eixo X, a largura correspondente ao eixo Y e o valor de cor do pixel correspondente ao eixo Z. Considerando uma unidade de pixel ou de cor equivalente a 1 metro, um fator de escala (2, 3, 1) representará um terreno com 512 metros de largura, 768 metros de comprimento e até 256 metros de altura. O mapa de relevo pode ainda ser replicado infinitamente lado a lado, gerando terrenos de tamanho infinito criados dinamicamente.

Vale ressaltar que o mapa de relevo da FIG. 5.8 (a) serve apenas para determinar a construção geométrica do terreno. A textura do terreno é aplicada separadamente, e pode-se utilizar qualquer imagem que contenha uma textura, como a da FIG. 5.8 (b). As FIGs. 5.9, 5.10 e 5.11 apresentam o terreno em sua forma aramada (a) e texturizada (b), com diferentes fatores de escala para a altitude (Z). Uma observação importante é que, se for necessário representar terrenos que possuam pontos com mais de 256 metros de altitude, o fator de escala para a altitude deverá ser maior do que 1. Quando o fator de escala é igual a 1, como na FIG. 5.11, os pontos totalmente brancos de um mapa de relevo (como ocorre com alguns pontos da FIG. 5.8 (a), por exemplo) correspondem a pontos com 256 metros de altitude. Apesar de permitir a representação de terrenos com pontos de grande altitude, essa técnica tem um efeito colateral. A discrepância entre dois pontos será sempre proporcional ao maior valor de altitude do terreno dividido por 256. Por exemplo, se o maior valor de altitude for 1024 metros, a diferença mínima de altitude entre dois pontos será de 4 metros.

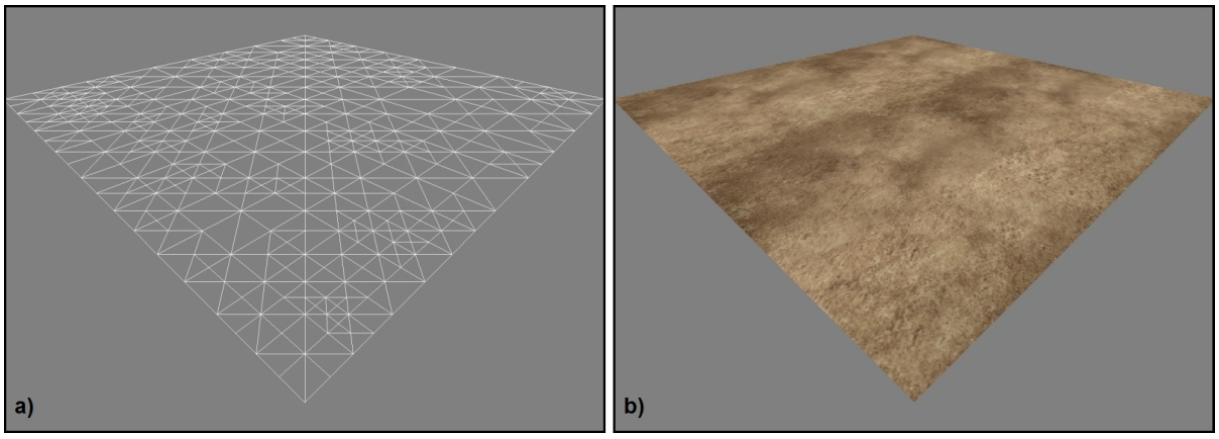

FIG. 5.9 – Terreno aramado (a) e texturizado (b) com fator de escala Z = 0.

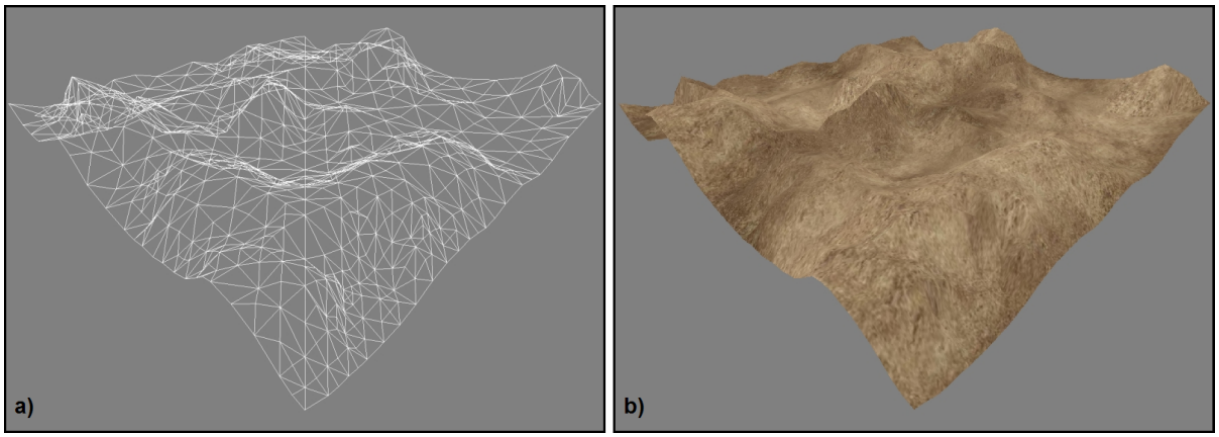

FIG. 5.10 – Terreno aramado (a) e texturizado (b) com fator de escala Z = 0,5.

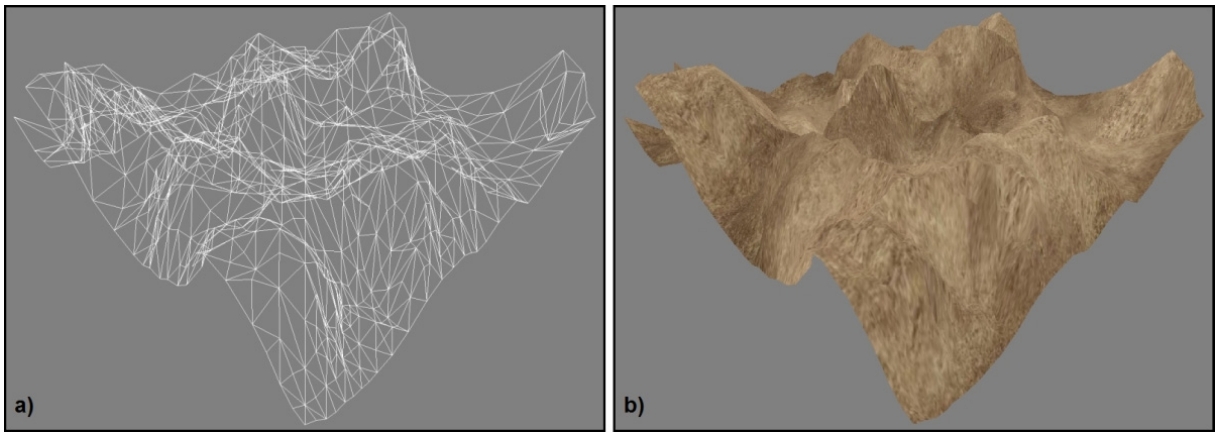

FIG. 5.11 – Terreno aramado (a) e texturizado (b) com fator de escala Z = 1.

A aplicação da textura pode ser feita de diversas formas. Pode-se aplicar uma textura pequena e repeti-la por todo o terreno, criando um padrão, ou pode-se determinar quantas células do terreno a imagem da textura vai ocupar. Com uma foto aérea de uma floresta, por exemplo, é possível aplicá-la como textura e causar uma aparência realista. Apesar de parecer que a textura do terreno consiste em uma única imagem inteira, a imagem da textura é replicada lado a lado, em todo o terreno, para abranger a área total desse.

5.2.3 A RENDERIZAÇÃO DE OBJETOS GENÉRICOS

A criação de um ambiente virtual não consiste apenas na renderização de um terreno, com seu relevo e textura. Para que um ambiente virtual seja semelhante a um ambiente real ele deve conter, além do terreno, os objetos estáticos do mundo real. Se for preciso simular uma missão de reconhecimento de uma pista de pouso, por exemplo, uma pista de pouso deverá estar presente no ambiente de simulação. Contudo, esses objetos geralmente são mais complexos, sendo constituídos, na maioria das vezes, de centenas ou até milhares de polígonos.

A criação de objetos 3D mais complexos, utilizando apenas a síntese de imagens a partir de vértices e formação de polígonos, requer bastante trabalho. Uma forma de reduzir esse trabalho e facilitar a adição de objetos estáticos ao ambiente virtual é através da importação de objetos 3D criados em ferramentas de modelagem existentes no mercado. Essas ferramentas geralmente oferecem muitos recursos, além de modelos pré-existentes, para criação de novos objetos tridimensionais. Dentre os principais softwares comerciais para modelagem 3D estão o *3D Studio Max*, o *Maya* e o *Corel Bryce*. Algumas ferramentas gratuitas também estão disponíveis, como o *Google Sketchup*, porém, com menos recursos.

A biblioteca OpenGL possui funções que facilitam a importação de arquivos contendo um objeto tridimensional, desde que este arquivo esteja em um formato padrão (geralmente com a extensão .obj ou .3ds). Outra questão favorável é que a internet é um grande repositório de objetos 3D gratuitos. Encontra-se desde aviões F-15 até castelos medievais.

Como um dos objetivos do simulador é fornecer subsídios visuais para o desenvolvimento de algoritmos baseados em imagens, a importação de objetos tridimensionais é uma função presente no simulador. A FIG. 5.12 mostra um ambiente fechado com alguns objetos estáticos importados para o ambiente a partir de arquivos de objetos 3D obtidos na internet.

FIG. 5.12 – Ambiente fechado com objetos 3D estáticos e um dirigível.

5.3 A OFICINA DE CONSTRUÇÃO DE DIRIGÍVEIS

No simulador desenvolvido para o projeto AURORA (RAMOS *et al.*, 1999), o dirigível virtual simulado é o AS-800, o mesmo utilizado na prática. Caso seja necessário simular outro tipo de dirigível no simulador do AURORA, as características do novo dirigível terão de ser informadas via programação. Uma característica importante do problema geral abordado nesta dissertação, e que o diferencia dos problemas já abordados, é a utilização de equipes

compostas por dirigíveis heterogêneos. Essa heterogeneidade diz respeito principalmente às características físicas do dirigível e ao aparato tecnológico embarcado.

Para facilitar a simulação de veículos heterogêneos sem a necessidade de alterar o código fonte do simulador, ou seja, de programar, o simulador possui uma oficina virtual para a construção do dirigível. É possível configurar desde as características físicas externas e os coeficientes dinâmicos adimensionais do dirigível até os tipos de sensores presentes. As subseções a seguir apresentam como são criadas as partes que compõem um dirigível.

5.3.1 A CONFIGURAÇÃO DO ENVELOPE

Com relação ao envelope de gás (ou invólucro), é possível configurar suas dimensões, a massa, o gás interno (hélio ou hidrogênio), o coeficiente de arrasto e a forma do envelope (elipsóide ou "gota"). O volume do envelope é automaticamente calculado, mas pode ser alterado manualmente pelo usuário. A posição com relação ao SCL também pode ser determinada, mas apenas para fins de design, pois posteriormente os componentes são reposicionados de acordo com o centro de gravidade computado pelo simulador. A FIG. 5.13 mostra a janela de configuração do envelope.

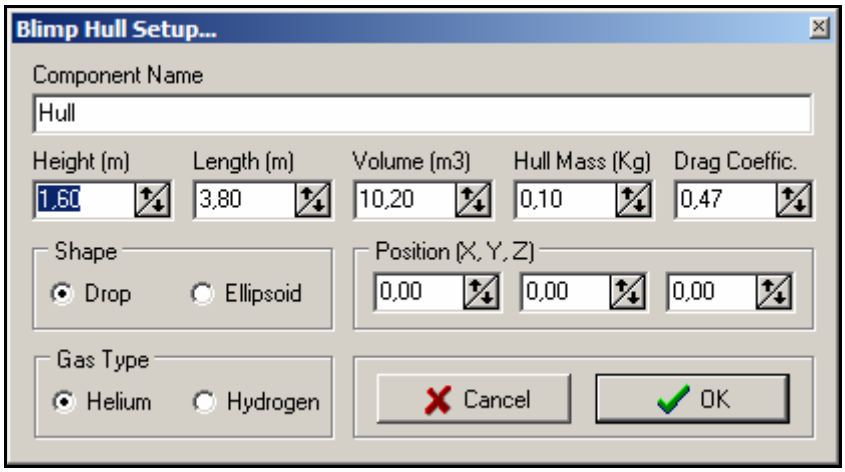

FIG. 5.13 – Janela de configuração do envelope de gás do dirigível.

O envelope é o primeiro componente que deve ser configurado ao se criar um novo dirigível. O volume de gás suportado pelo envelope é que vai determinar a quantidade de componentes que poderão ser embarcados no dirigível. A FIG. 5.14 apresenta algumas configurações de envelope. A FIG. 5.14 (a) mostra um envelope de um dirigível de pequeno porte, com comprimento de 3,8 metros e diâmetro de 1,6 metros, com um volume aproximado

de 10,2 m^3. Na FIG. 5.14 (b), é mostrado um envelope também com comprimento de 3,8 metros, mas com diâmetro de 2 metros, resultando em um volume aproximado de 16 m^3. Por fim, a FIG. 5.14 (c) apresenta um envelope em formato de elipsóide, como ocorre em alguns modelos de dirigíveis de médio porte. O envelope tem comprimento de 12 metros e diâmetro de 2,6 metros, resultando em um volume de 86 m^3.

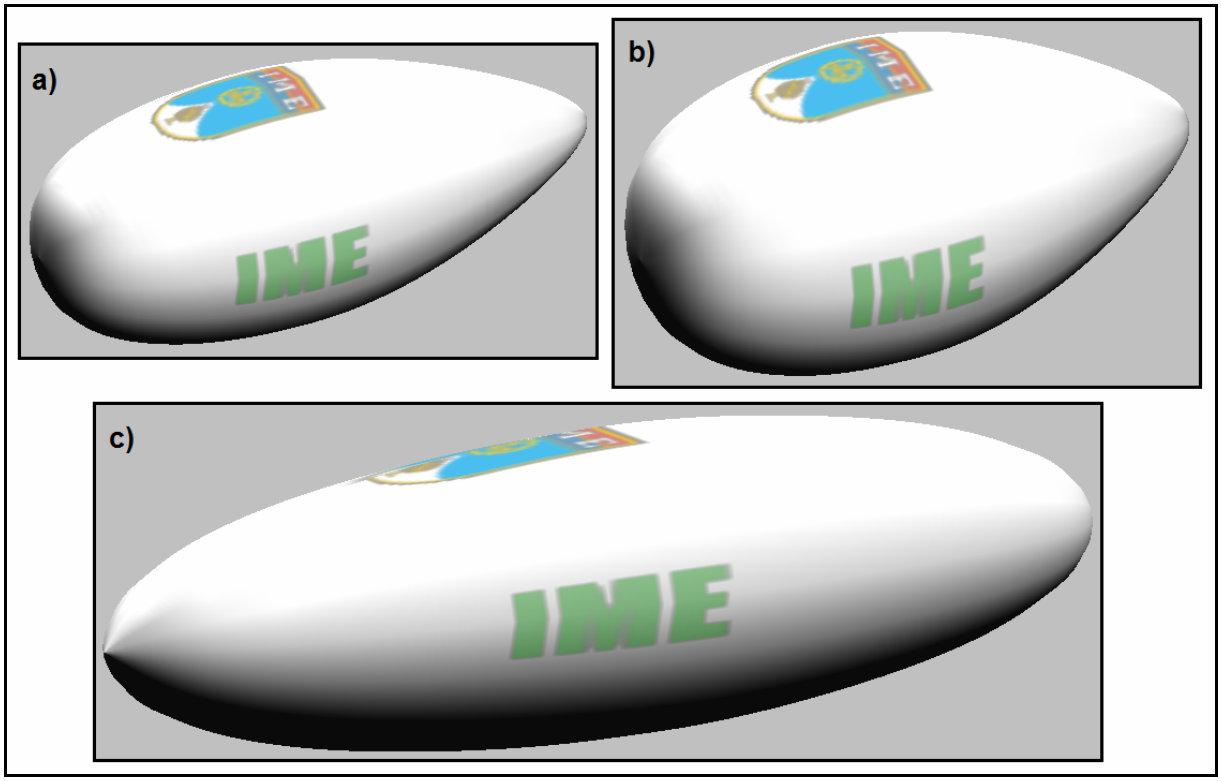

FIG. 5.14 – Alguns tipos de envelopes criados na oficina virtual.

5.3.2 A CONFIGURAÇÃO DA GÔNDOLA

Outro componente configurável é a gôndola. A gôndola é um componente simples, que não influencia deliberadamente na movimentação do dirigível. Por isso, ela possui apenas as configurações comuns a todos os componentes. É possível configurar a posição relativa ao SCL, a massa, as dimensões e o coeficiente de arrasto. A FIG. 5.15 mostra a janela de configuração da gôndola. Os componentes embarcados, como GPS, câmeras etc., geralmente são colocados dentro da gôndola. Contudo, a massa desses componentes não é adicionada à massa da gôndola, pois suas massas individuais são configuradas em outra janela, como é apresentado mais adiante.

A configuração das dimensões da gôndola não implica na configuração automática de sua massa e de seu coeficiente de arrasto. Esses valores devem ser informados diretamente pelo usuário. Caso o usuário esteja criando um dirigível baseado em um modelo real, a massa da gôndola pode ser facilmente determinada através de uma balança. Já o coeficiente de arrasto não é tão simples de ser determinado. Uma recomendação é que o usuário utilize um coeficiente de arrasto conhecido, de algum sólido regular semelhante à gôndola.

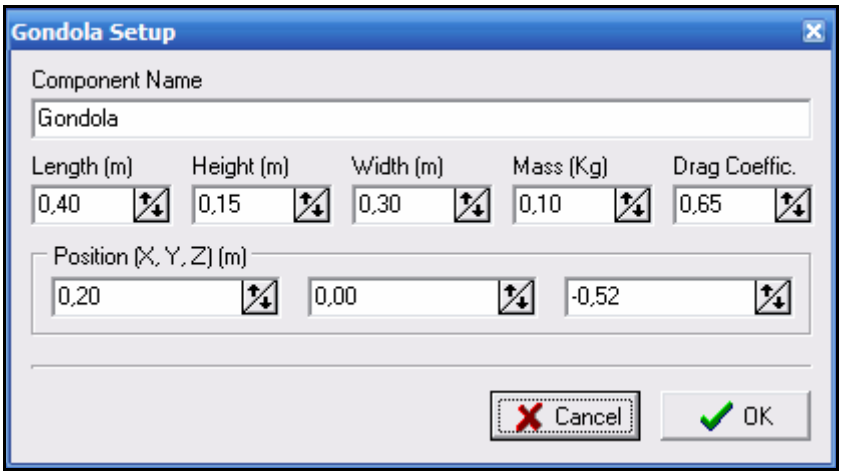

FIG. 5.15 – Janela de configuração da gôndola do dirigível.

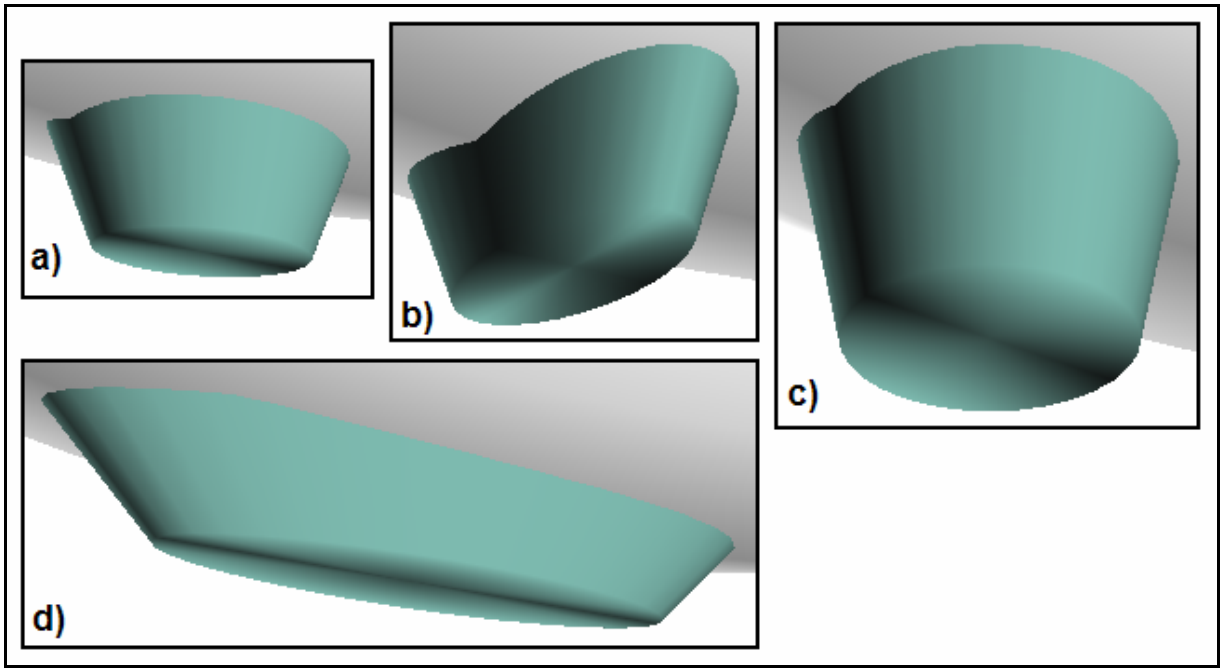

FIG. 5.16 – Diferentes configurações de gôndola feitas na oficina virtual.

A FIG. 5.16 apresenta algumas possíveis configurações de gôndola. A FIG. 5.16 (a) e (d) apresentam gôndolas longitudinais, que certamente terão coeficientes de arrasto menores do que os coeficientes das gôndolas (b) e (c).

5.3.3 A CONFIGURAÇÃO DOS PROPULSORES DE GÔNDOLA

Os propulsores da gôndola também podem ser configurados, sendo que a mesma configuração vale para os propulsores da esquerda e da direita. Com relação à posição, apenas a altura e a posição longitudinal podem ser configuradas. Assim como nos componentes comuns, a massa e o coeficiente de arrasto também são passíveis de configuração. Algumas configurações peculiares aos propulsores são: o tamanho do eixo, que define a distância lateral do dirigível ao propulsor; a potência máxima, em rotações por segundo; o consumo de energia (ou bateria); e, o coeficiente de propulsão da hélice. Opcionalmente, os propulsores da gôndola podem ser vetorizáveis, ou seja, o eixo de sustentação, paralelo ao eixo latitudinal (Y) do dirigível, pode rotacionar. Mesmo que os propulsores não sejam vetorizáveis durante o vôo, eles podem estar com uma angulação fixa diferenciada. Essa angulação fixa também pode ser configurada. A FIG. 5.17 mostra a janela de configuração dos propulsores de gôndola.

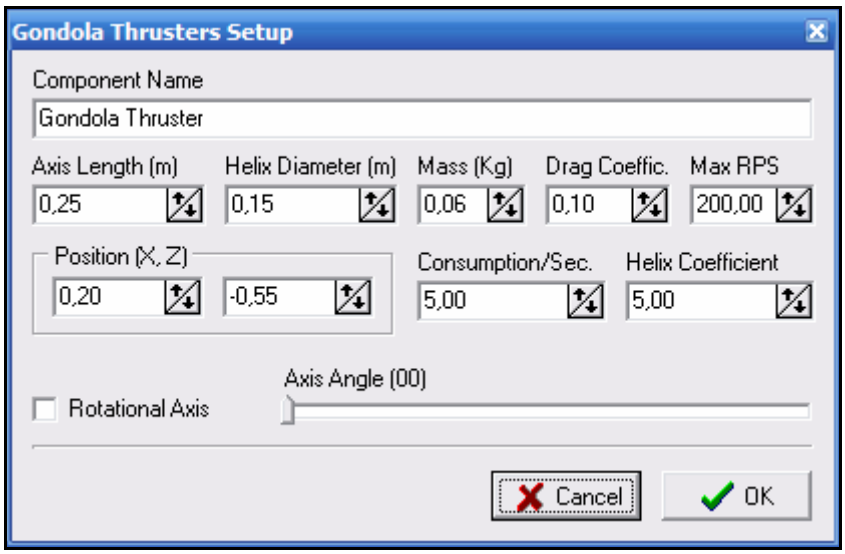

FIG. 5.17 – Janela de configuração dos propulsores de gôndola do dirigível.

As configurações do diâmetro da hélice, do coeficiente de propulsão da hélice e da quantidade máxima de rotações por segundo é que determinam a força que o propulsor será capaz de exercer durante o vôo. A FIG. 5.18 mostra algumas configurações de propulsores de

gôndola. Entre a FIG. 5.18 (a) e a FIG. 5.18 (b) é possível notar uma diferença de tamanho do eixo de sustentação dos propulsores. Essa configuração pode fazer diferença se o dirigível puder ser rotacionado aplicando-se forças diferentes, ou até mesmo contrárias, nos propulsores esquerdo e direito, pois quanto maior a distância do propulsor com relação ao centro de gravidade do dirigível, maior será o torque gerado. A FIG. 5.18 (c) mostra um propulsor com uma hélice de diâmetro superior às hélices dos propulsores da FIG. 5.18 (a) e (b).

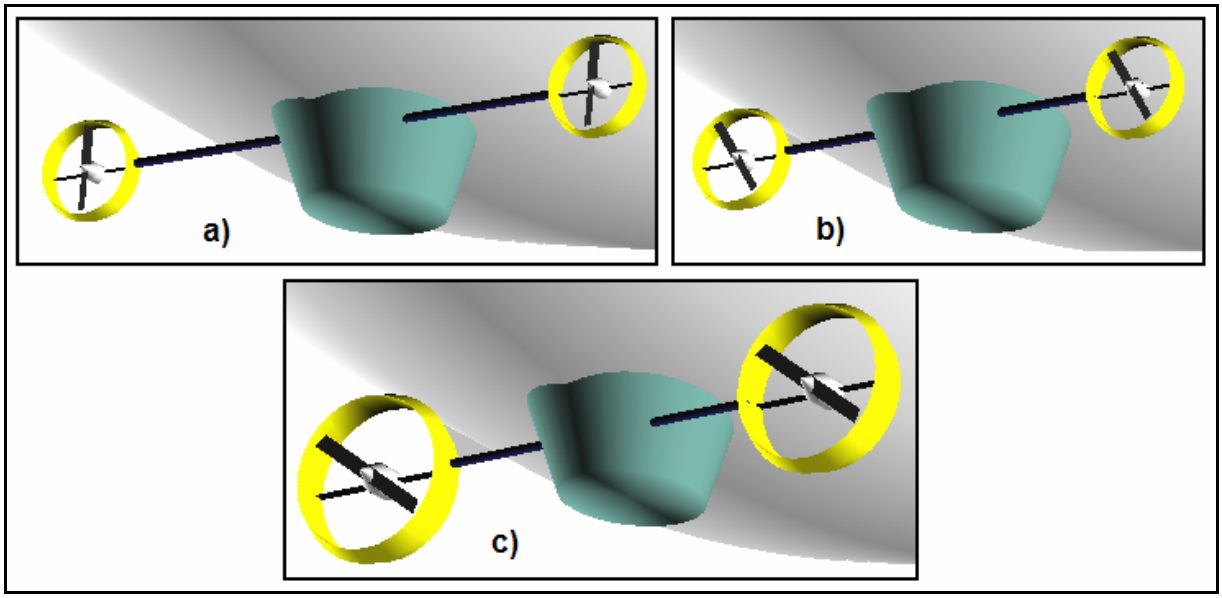

FIG. 5.18 – Diferentes configurações para os propulsores de gôndola.

5.3.4 A CONFIGURAÇÃO DOS LEMES DE CAUDA

Como visto na subseção 5.1.3, o dirigível possui componentes exclusivamente aerodinâmicos. Esses componentes são os lemes de cauda, que também podem ser configurados na oficina de construção de dirigíveis. Na atual versão do simulador, a posição dos lemes não pode ser configurada, sendo que eles ficam sempre em formato de cruz, no fim da cauda do dirigível. O comprimento e a largura dos lemes são configuráveis, bem como sua massa e seu coeficiente aerodinâmico, que equivale ao coeficiente de arrasto dos outros componentes. Para simplificar, as configurações são aplicadas igualmente aos quatro lemes. A FIG. 5.19 mostra a janela de configuração dos lemes de cauda.

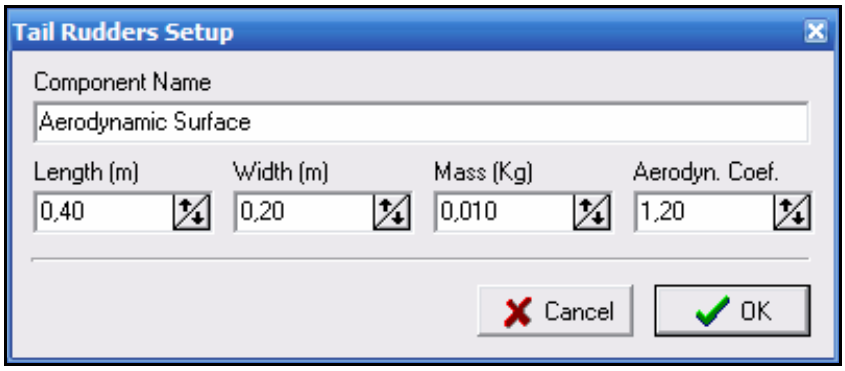

FIG. 5.19 – Janela de configuração dos lemes de cauda do dirigível.

A FIG. 5.20 mostra algumas configurações de lemes de cauda. A FIG. 5.20 (a) mostra um leme de cauda discreto, que terá uma influência pequena na movimentação do dirigível. Na FIG. 5.20 (b), os lemes são mais compridos, exercendo maior influência na movimentação do dirigível. Por fim, a FIG. 5.20 (c) exibe um leme de tamanho considerável, e que certamente exercerá grande resistência a guinadas e demais movimentos angulares do dirigível.

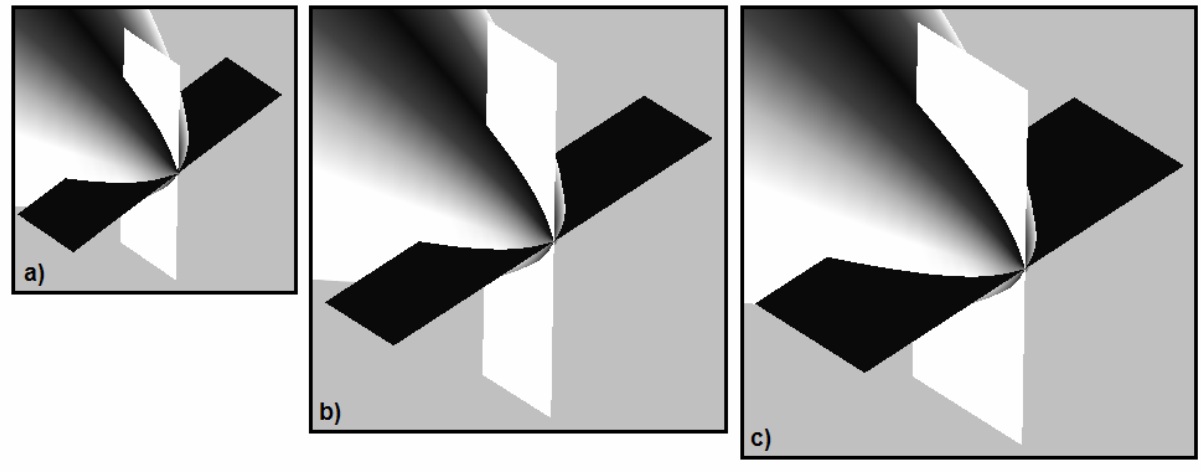

FIG. 5.20 – Diferentes configurações dos lemes de cauda do dirigível.

5.3.5 A CONFIGURAÇÃO DO PROPULSOR DE CAUDA

O propulsor de cauda também pode ser configurado. As configurações são praticamente as mesmas dos propulsores da gôndola. A única diferença é que o propulsor de cauda não tem a opção de ser vetorizável. A FIG. 5.21 apresenta a janela de configuração do propulsor de cauda. Em alguns dirigíveis reais, o propulsor de cauda fica embutido no leme de cauda vertical, superior ou inferior. Na versão atual do simulador, o leme tem a opção de ficar apenas no fim do envelope, conforme mostra a FIG. 5.22. Na FIG. 5.22 (a) é apresentado um

propulsor de tamanho relativamente pequeno. Propulsores de cauda muito grandes, como o da FIG. 5.22 (b), geralmente são muito pesados, fazendo o dirigível levantar a parte dianteira. Se a frente do dirigível tiver uma inclinação natural muito grande, os propulsores de gôndola deverão estar vetorizados para baixo para que o dirigível navegue mantendo uma altitude. Também, a aerodinâmica fica prejudicada, pois a frente do dirigível passa a incluir parte da porção inferior frontal do envelope, que, por sua vez, aumenta a superfície de incidência do vento e, conseqüentemente, aumenta também o arrasto aerodinâmico.

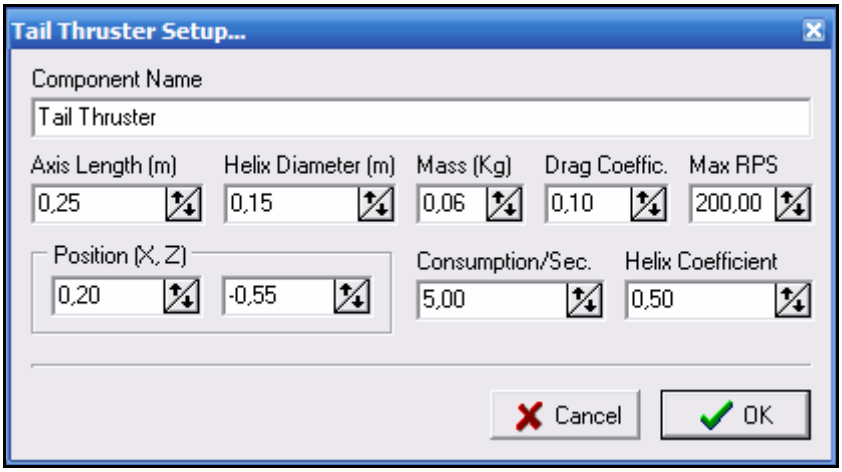

FIG. 5.21 – Janela de configuração do propulsor de cauda do dirigível.

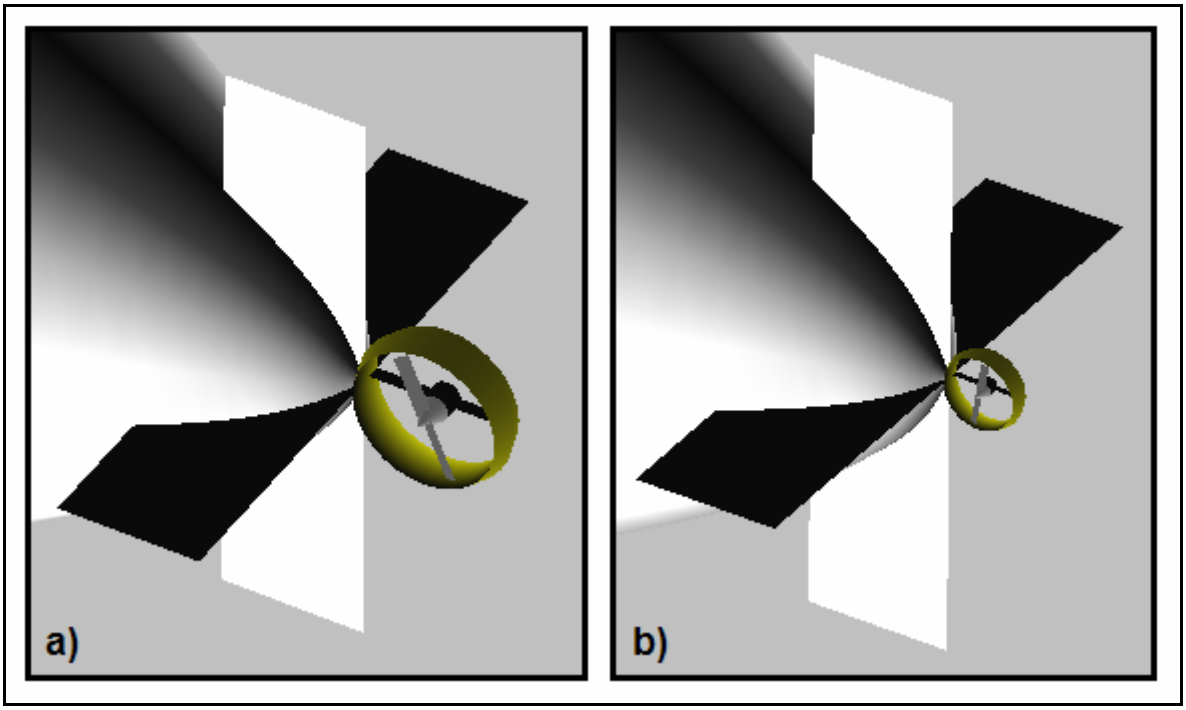

FIG. 5.22 – Diferentes configurações do propulsor de cauda do dirigível.

5.3.6 A CONFIGURAÇÃO DOS COMPONENTES EMBARCADOS

Os principais componentes que podem ser embarcados (sensores) também podem ser configurados. Eles podem estar ou não presentes no dirigível e, no caso da câmera, há a opção dela ser monócula ou estereoscópica, ou seja, ter uma ou duas câmeras. Na atual versão do simulador, os sensores passíveis de simulação são a câmera, o giroscópio, o inclinômetro, o acelerômetro, o altímetro e o GPS, cujos detalhes são apresentados na subseção 3.2.1. A FIG. 5.17 mostra a janela de configuração dos componentes embarcados.

A adição de todos os componentes, exceto os sensores, requer a definição das suas posições relativas ao sistema de coordenadas local do dirigível. Entretanto, durante a construção, a definição da posição configurada pelo usuário está relacionada ao centro volumétrico do envelope.

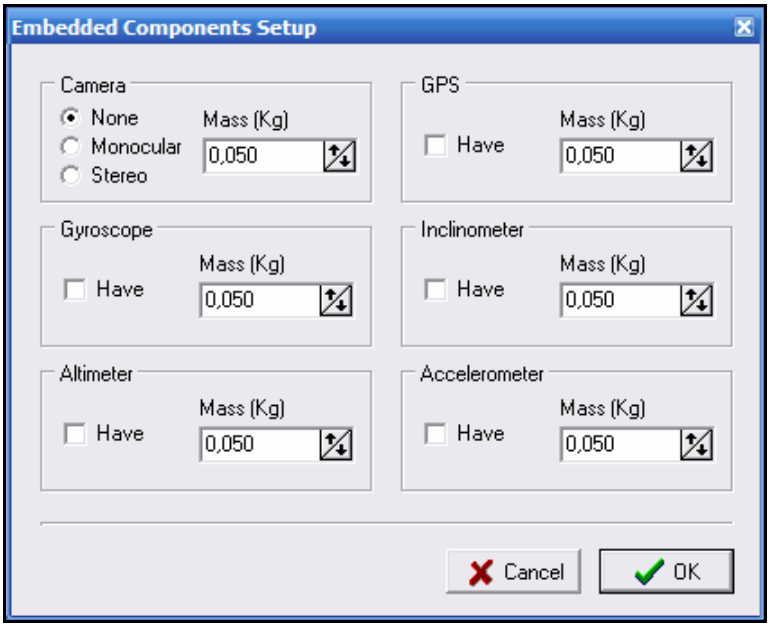

FIG. 5.23 – Janela de configuração dos equipamentos embarcados do dirigível.

Como os torques são calculados de acordo com a distância dos componentes ao centro de gravidade, após adicionar os componentes, deve-se determinar o centro de gravidade do dirigível através do comando apropriado. A FIG. 5.18 apresenta a janela da oficina de construção de dirigíveis, exibindo um dirigível já construído. Na barra de ferramentas, o nono botão da esquerda para a direita (com rótulo CG, de *Center of Gravity*), ajusta as coordenadas

locais dos componentes de acordo com o centro de gravidade do dirigível. O oitavo botão retorna as coordenadas às configurações determinadas durante a criação dos componentes.

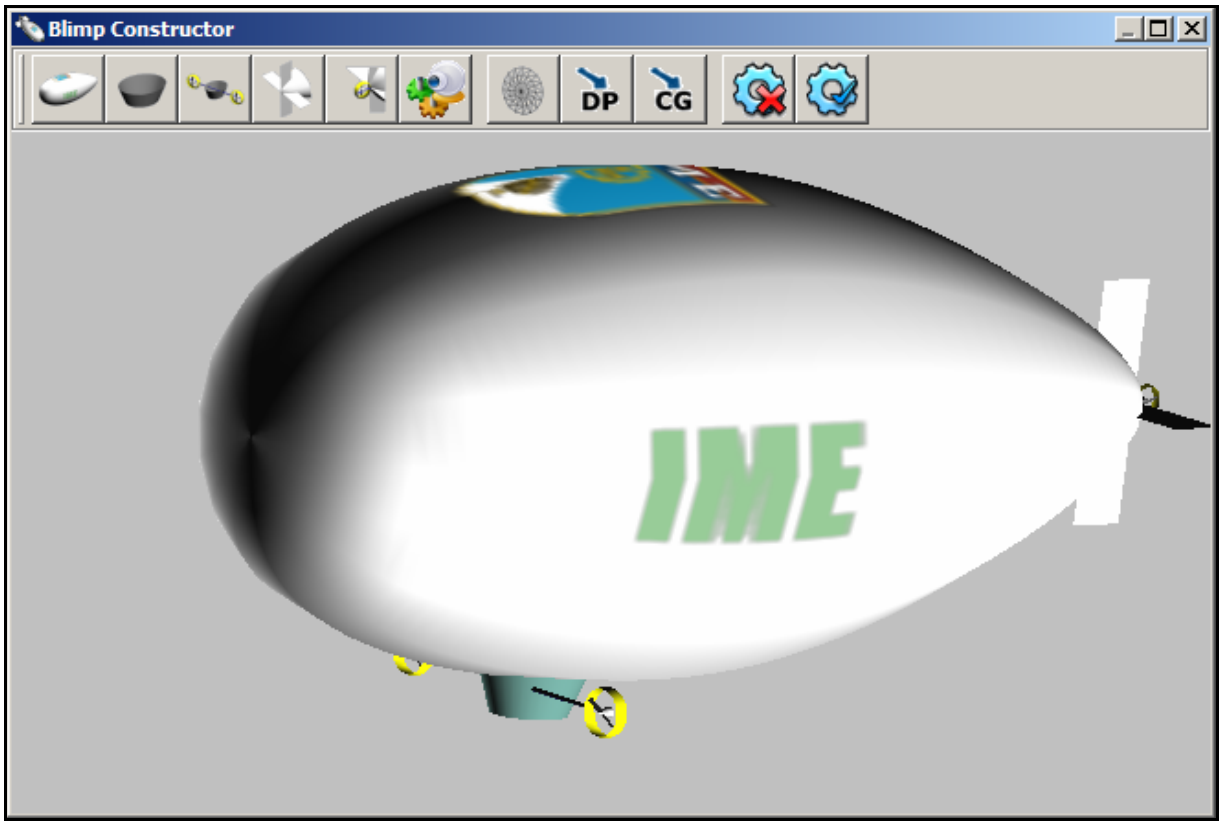

FIG. 5.24 – Janela de criação do dirigível.

5.4 A OPÇÃO DE EXTENSÃO DO SIMULADOR

Um dos temas de maior interesse da Robótica atualmente é o desenvolvimento de robôs inteligentes. A inteligência pode estar em qualquer atividade que possa ser realizada de forma autônoma. A navegação autônoma, por exemplo, é um tema desafiador para a Robótica Móvel. Um sistema de navegação inteligente pode permitir que um veículo navegue de forma autônoma e em segurança por um ambiente, evitando colisões e decidindo qual a melhor trajetória a percorrer para a tarefa que está para realizar, mesmo que obstáculos móveis estejam no ambiente.

Um dos requisitos para a continuidade do presente trabalho é que o desenvolvimento de sistemas inteligentes no simulador seja fácil e flexível, preferencialmente de forma que não obrigue a alteração direta do código fonte do simulador e sua recompilação. Uma solução

plausível é disponibilizar algum tipo de interface para que o simulador possa se comunicar com sistemas externos, desenvolvidos até mesmo em outras linguagens. Em termos técnicos, deve-se providenciar alguma forma de comunicação entre processos.

Uma forma conhecida de se trocar mensagens entre programas é o compartilhamento de memória, através de *pipes* ou de *dlls*. Contudo, essa solução é interessante quando os processos residem na mesma máquina. Quando os processos são executados em estações diferentes, a utilização dessa técnica é desaconselhada, pois ela depende fortemente do sistema operacional utilizado nas estações.

Outra possibilidade é a troca de mensagens via soquetes[2]. Um soquete é usado em ligações de redes de computadores para criar um elo bidirecional de comunicação entre dois programas. A interface padronizada de soquetes surgiu originalmente no sistema operacional Unix BSD (*Berkeley Software Distribution*). Um soquete também pode ser visto como uma abstração computacional que mapeia diretamente a uma porta de transporte (TCP ou UDP) e a um endereço de rede. Com esse conceito é possível identificar unicamente um aplicativo ou servidor na rede de comunicação IP. Em documentos de RFC (*Request for Comments*) relacionados ao TCP ou UDP, um soquete em um computador é definido como a combinação de um endereço IP, um protocolo e o número da porta do protocolo.

Depois de estabelecida uma comunicação via soquete, é possível trocar dados entre as aplicações pela rede. Dessa forma, pode-se estabelecer um protocolo em nível de aplicação para determinar os tipos das mensagens que são trocadas.

A segunda opção (soquetes) foi a escolhida para criar a interface de comunicação do simulador com outros programas. Um protocolo em nível de aplicação foi implementado, permitindo a um cliente remoto conectado ao simulador solicitar dados de sensores e comandar os atuadores dos dirigíveis. Assim, um sistema inteligente, para controle e navegação, por exemplo, pode ser desenvolvido de forma independente em qualquer linguagem que ofereça suporte à programação de soquetes. Algumas linguagens que suportam a manipulação de soquetes são: C, C++, Java, C#, Object Pascal, dentre outras.

---

[2] Do inglês *sockets*.

5.4.1 O PROTOCOLO DE ACESSO REMOTO

Para estabelecer a comunicação entre duas aplicações via soquetes, é preciso trocar mensagens. Entretanto, essas mensagens devem obedecer a um padrão, para que ambas as aplicações, cliente e servidora, interpretem corretamente os comandos. Para isso, foi estabelecido um protocolo em nível de aplicação para realizar a troca de mensagens entre o simulador, no papel de servidor, e uma aplicação cliente qualquer que queira utilizar dados do simulador para, por exemplo, estendê-lo com algoritmos inteligentes.

O protocolo estabelecido é básico, consistindo em comandos de texto que seguem um esquema de passagem de parâmetros, dependendo do tipo de comando dado. Um requisito básico do servidor de simulação é que ele seja capaz de fornecer dados específicos de um dirigível ou do ambiente, como dados do GPS (posição), do inclinômetro (orientação), imagens das câmeras embarcadas nos dirigíveis, o mapa de relevo do ambiente etc. Além de comandos para obtenção de dados, há também comandos instrutivos, para acionar propulsores, definir uma rota etc. Os comandos aceitos pelo servidor de simulação – que compõem o protocolo de comunicação entre o servidor de simulação e uma aplicação cliente – são apresentados a seguir.

Comandos para Obtenção de Dados

- BLPCOUNT: obtém a quantidade de dirigíveis ativos na simulação. Não é necessário nenhum outro parâmetro para esse comando. O simulador retorna um único valor numérico contendo a quantidade de dirigíveis ativos.

- BLPSNAMS: obtém uma lista com os nomes dos dirigíveis ativos na simulação. Não é necessário nenhum outro parâmetro para esse comando. O simulador retorna uma lista de nomes, onde cada nome se encontra em uma linha dessa lista.

- BLMPOSIT: obtém a posição global de um dirigível, simulando um GPS. É necessário informar o número do dirigível que se quer consultar. O simulador retorna três números, que representam as coordenadas de latitude, longitude e altitude do dirigível.

- BLMPORTN: obtém a orientação do dirigível com relação ao sistema de coordenadas do ambiente, como um inclinômetro. É preciso informar o número do dirigível que se

quer consultar. O simulador retorna três números, que representam quantos graus o corpo está rotacionado em torno dos eixos X, Y e Z do SCA, em graus.

- BLMPALTM: obtém o valor da altitude do dirigível perpendicularmente ao solo, como um altímetro. É preciso informar o número do dirigível que se quer consultar. O simulador retorna um único número correspondente à altitude do dirigível.

- BPCMRIMG: obtém a imagem da câmera embarcada do dirigível. É preciso informar o número do dirigível e se a imagem é da câmera esquerda, direita ou central, quando monóculo. O simulador retorna um *stream* de dados contendo a imagem da câmera solicitada.

- BPBATTRY: obtém a quantidade de bateria (combustível) restante no dirigível. É preciso informar o número do dirigível. O simulador retorna um valor numérico correspondente ao percentual de bateria restante.

- AMBHTMAP: obtém o mapa de relevo de um ambiente aberto. Não é preciso passar parâmetros, mas funciona somente em simulações de ambientes abertos. O simulador retorna um fluxo de dados contendo a imagem do mapa de relevo do ambiente.

Comandos Instrutivos

- BPTHRSTL: altera a potência do propulsor esquerdo. É preciso informar o número do dirigível e a nova potência do propulsor. O simulador não retorna dado para esse comando.

- BPTHRSTR: idêntico a BPTHRSTL, mas para o propulsor direito.

- BPTHRSTT: idêntico a BPTHRSTL, mas para o propulsor de cauda.

- BPTHRROT: altera o ângulo de vetorização dos propulsores da gôndola. É preciso informar o número do dirigível e o novo ângulo de vetorização. O simulador não retorna dado para esse comando.

- BPPATHND: informa um novo nó (coordenada), parte de um caminho que deve ser seguido pelo dirigível. É preciso informar o número do dirigível, os três componentes da

coordenada do nó, a velocidade até o próximo nó e o ângulo que será rotacionado até ele. O simulador retorna a quantidade total de nós após essa inserção.

- BPPATHFW: informa a um dirigível que ele deve seguir uma trajetória pré-composta por comandos BPPATHND. É preciso informar o número do dirigível que percorrerá a trajetória. Quando o dirigível chega ao objetivo, o simulador informa ao cliente o momento exato da chegada e o índice do dirigível.

- BPPATHCL: exclui todos os nós de um caminho a ser seguido pelo dirigível. É preciso informar o número do dirigível que será afetado. O simulador não retorna dado para esse comando.

No Capítulo 6, são apresentados dois sistemas para determinação de trajetórias que se comunicam com o simulador utilizando a interface via soquetes. Primeiramente, o sistema externo (cliente) solicita o mapa do terreno ao simulador (servidor) e a posição do dirigível. Após o cômputo do caminho do local de partida do dirigível até um ponto objetivo, o sistema externo informa ao simulador o caminho computado e envia um comando para que o dirigível siga esse caminho. Os sistemas foram implementados utilizando duas abordagens distintas. Um dos sistemas foi implementado com o algoritmo A*, que é um algoritmo clássico da IA. O outro sistema foi implementado com o algoritmo de Aprendizado por Reforço conhecido como *Q-Learning*.

5.5 TESTES E RESULTADOS OBTIDOS COM O SIMULADOR

Com o objetivo de validar a utilização do simulador, alguns testes foram realizados. Os testes consistem basicamente em avaliar alguns ambientes e sua aproximação visual com a realidade, além de analisar o desempenho, principalmente da parte gráfica e da utilização remota do simulador. A avaliação das imagens consiste basicamente em construir alguns ambientes e avaliar visualmente seu realismo. No entanto, os gráficos tridimensionais requerem muitos recursos computacionais para serem exibidos, e o desempenho pode ficar comprometido. Uma forma de analisar o desempenho é medir a quantidade de quadros por segundo que está sendo exibida em um instante de uma simulação em tempo real. Já a utilização remota é limitada pela largura de banda disponível no canal de comunicação entre a

aplicação cliente e a aplicação servidora. Ambos os testes foram feitos separadamente e divididos nas duas próximas subseções que seguem.

5.5.1 AVALIAÇÃO DAS IMAGENS E DO DESEMPENHO GRÁFICO

A avaliação das imagens consiste na modelagem de alguns ambientes e na avaliação visual desses ambientes. A seguir, são apresentadas figuras de alguns ambientes modelados no simulador, representando aproximações de ambientes reais de atuação dos dirigíveis.

A FIG. 5.25 apresenta um ambiente urbano, construído como um ambiente fechado composto por vários objetos estáticos importados para o ambiente de simulação. Os objetos importados estão todos disponíveis no próprio simulador e podem ser facilmente inseridos. O ambiente da FIG. 5.25 possui 1.000 metros de comprimento, 1.000 metros de largura e 300 metros de altura. Apesar de ser um ambiente fechado, a textura das paredes e do teto dão um aspecto de ambiente aberto, pois são utilizadas imagens de nuvens. A FIG. 5.26 mostra as texturas utilizadas no ambiente de simulação.

FIG. 5.25 – Ambiente urbano com 3 dirigíveis sobrevoando as ruas.

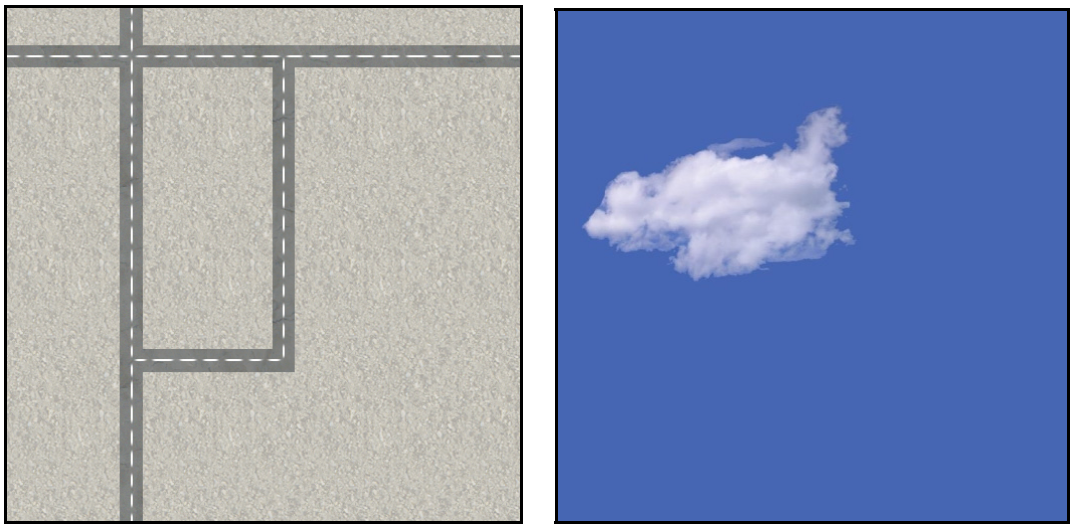

FIG. 5.26 – Texturas usadas no ambiente de simulação urbano da FIG. 5.25.

Na FIG. 5.26, à esquerda está a textura do chão, que consiste em um piso de terra com asfalto que, quando colocado lado a lado, forma um conjunto de estradas interligadas. À direita está a textura do teto e das paredes, que consiste em um fundo azul como uma nuvem, para dar um aspecto celeste. As texturas do chão, paredes e teto possuem, respectivamente, densidades iguais a 4, 4 e 5. A densidade determina o tamanho que a figura terá ao ser usada como textura, determinando a granularidade da textura na superfície. Os objetos estáticos que compõem o ambiente, bem como suas características, são mostrados na TAB. 5.1.

TAB. 5.1 – Objetos inseridos no ambiente da FIG. 5.19.

Objeto	Posição (X, Y, Z)	Rotação (graus)	Fator de Escala
Buildings	(-100; 20; 40,9)	0	0,03
Buildings	(15; 60; 40,9)	90	0,03
Buildings	(-70; 140; 40,9)	90	0,03
Buildings	(150; 20; 40,9)	0	0,03
Buildings	(135; 195; 40,9)	90	0,03
Buildings	(20; 225; 40,9)	0	0,03
Buildings	(275; 225; 40,9)	0	0,03
Buildings	(200; 132; 40,9)	90	0,03
Buildings	(265; 60; 40,9)	90	0,03
Buildings	(-25; -70; 40,9)	0	0,03
Buildings	(225; -70; 40,9)	0	0,03
Buildings	(265; -185; 40,9)	90	0,03
Buildings	(30; -225; 40,9)	0	0,03
Buildings	(400; 400; 40,9)	0	0,03
Car	(180; -200; 0,9)	0	0,04
Car	(190; -200; 0,9)	0	0,04
Car	(200; -200; 0,9)	0	0,04
Car	(210; -200; 0,9)	0	0,04
Car	(220; -200; 0,9)	0	0,04
Car	(180; -190; 0,9)	0	0,04
Car	(190; -190; 0,9)	0	0,04

(Continuação da Tabela 5.1)

Objeto	Posição (X, Y, Z)	Rotação (graus)	Fator de Escala
Car	(200; -190; 0,9)	0	0,04
Car	(210; -190; 0,9)	0	0,04
Car	(220; -190; 0,9)	0	0,04
Car	(180; -180; 0,9)	0	0,04
Car	(190; -180; 0,9)	0	0,04
Car	(200; -180; 0,9)	0	0,04
Car	(210; -180; 0,9)	0	0,04
Car	(220; -180; 0,9)	0	0,04
Car	(10; -21; 0,9)	-90	0,04
Car	(60; -21; 0,9)	-90	0,04
Mansion	(-100; -100; 5)	0	0,04
Mansion	(130; 50; 5)	0	0,04
Mansion	(130; 80; 5)	0	0,04
Mansion	(130; 110; 5)	0	0,04
Mansion	(130; 140; 5)	0	0,04
Mansion	(135; -170; 5)	0	0,04
Mansion	(135; -200; 5)	0	0,04
Mansion	(90; -110; 5)	90	0,04
Mansion	(135; -90; 5)	0	0,04
Mountain House	(-150; -150; 5,5)	90	1,00
Mountain House	(40; -115; 5,5)	0	1,00
Mountain House	(0; -145; 5,5)	0	1,00
Mountain House	(-40; -145; 5,5)	0	1,00
Mountain House	(90; -80; 5,5)	0	1,00
Oil Tank	(-150; -100; 2,9)	0	0,25
Oil Tank	(-150; -80; 2,9)	0	0,25
Oil Tank	(190; -150; 2,9)	0	0,25
Oil Tank	(210; -150; 2,9)	0	0,25
Pub	(-135; -120; 3,7)	0	0,05
Pub	(45; -5; 3,7)	0	0,05
Pub	(130; -240; 3,7)	0	0,05
Square Shopping	(120; -150; 3)	90	0,02
Square Shopping	(-130; -150; 3)	90	0,02
Stadium Football	(225; 370; 23)	11	0,02
Tree	(120; -45; 7,5)	0	0,05
Tree	(120; -55; 7,5)	0	0,05
Tree	(120; -65; 7,5)	0	0,05
Tree	(120; -75; 7,5)	0	0,05
Tree	(120; -85; 7,5)	0	0,05
Tree	(120; -95; 7,5)	0	0,05
Tree	(120; -105; 7,5)	0	0,05
Tree	(120; -115; 7,5)	0	0,05
Tree	(120; -125; 7,5)	0	0,05
Tree	(120; -135; 7,5)	0	0,05
Tree	(120; -145; 7,5)	0	0,05
Tree	(120; -155; 7,5)	0	0,05
Tree	(80; 0; 7,5)	0	0,05
Truck	(0; -27; 2,6)	90	0,04
Wind Mill	(-130, -100, 10)	90	0,60
Wind Mill	(-130, -80, 10)	90	0,60

O cenário é composto por 71 objetos estáticos, sendo que alguns deles parecem ser mais de um objeto. O objeto *Buildings*, por exemplo, consiste em um conjunto de 5 prédios. Além de prédios, há como inserir diversos objetos no ambiente de simulação, como casas, estádios

de futebol, árvores, carros, caminhões, lojas, lanchonetes etc. Com a TAB. 5.1, é possível reconstruir um ambiente idêntico ao da FIG. 5.25.

A FIG. 5.27 mostra a visão da câmera embarcada de um dos dirigíveis apontando para um estádio de futebol. Como se pode observar, a possibilidade de se criar diferentes ambientes com um bom nível de semelhança com a realidade depende não só da renderização do terreno e de seu relevo em si, mas também dos objetos inseridos no ambiente de simulação.

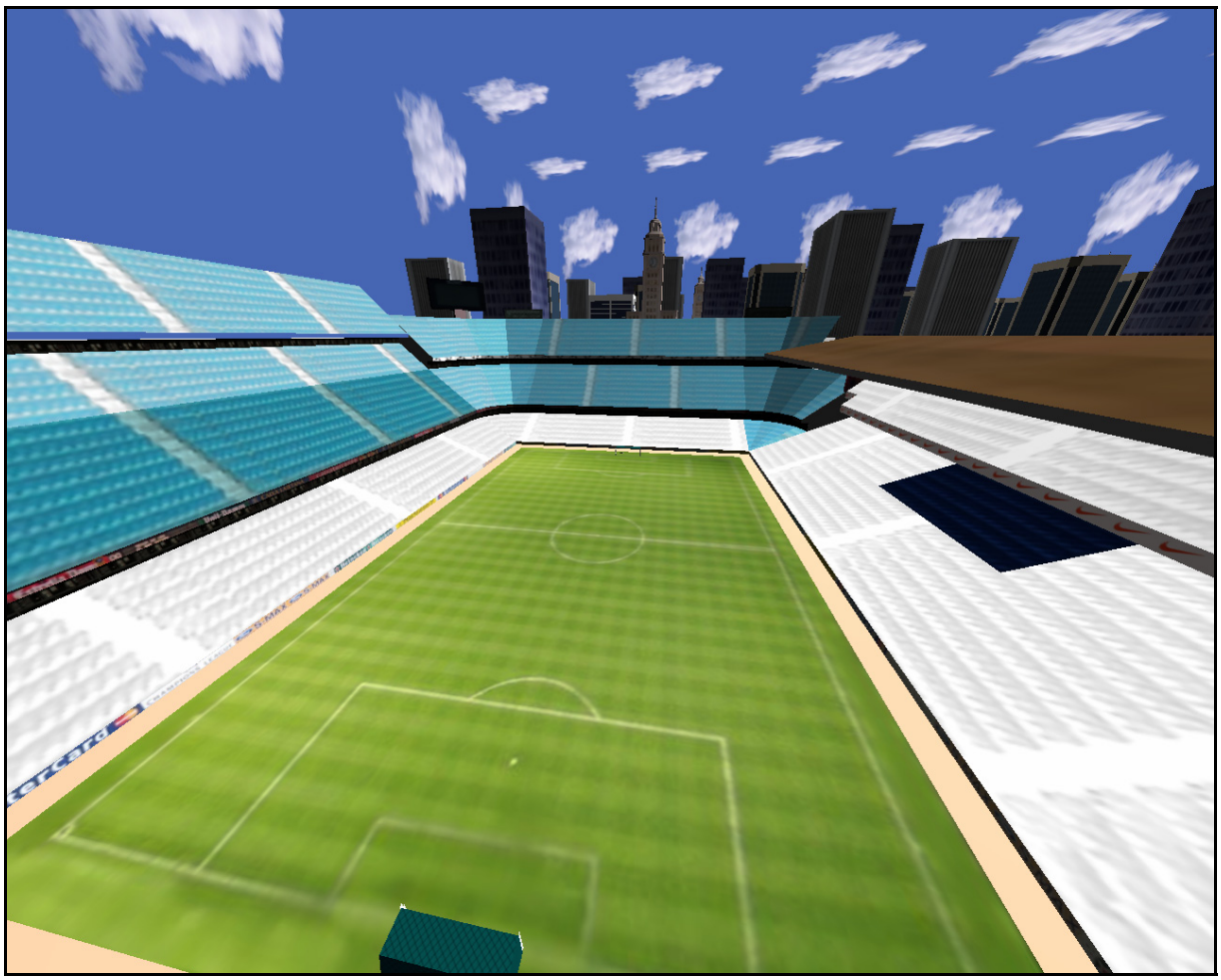

FIG. 5.27 – Imagem do estádio obtida a partir da câmera embarcada no dirigível.

A FIG. 5.28 mostra uma visão aérea do ambiente urbano criado a partir da TAB. 5.1. É possível diferenciar claramente os diferentes tipos de estruturas representados pelos objetos tridimensionais inseridos no ambiente de simulação. É possível distinguir árvores, diferentes prédios, casas, ruas, o estádio etc.

A importação de objetos tridimensionais para o ambiente de simulação aumenta o realismo, mas aumenta também o tempo de computação para renderizar as imagens. A TAB. 5.2 apresenta um teste de desempenho realizado com a simulação do ambiente urbano em questão, com todos os objetos da TAB. 5.1 ativados (visíveis).

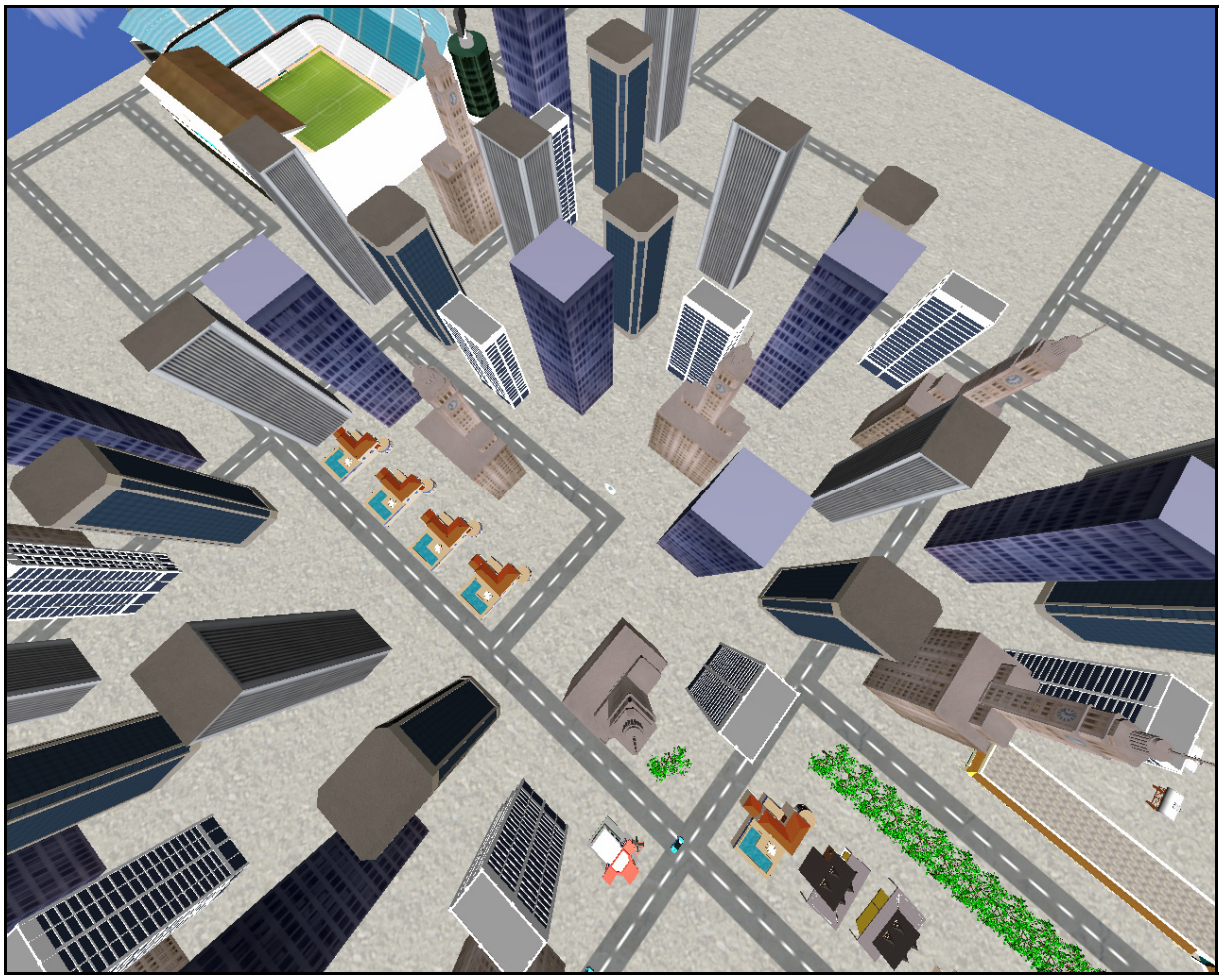

FIG. 5.28 – Visão aérea do ambiente urbano criado a partir da TAB. 5.1.

TAB. 5.2 – Avaliação do desempenho de renderização do ambiente urbano.

Qtde. Objetos	Qtde. Dirigíveis	Resolução	Quadros/Segundo
71	3	320 x 240	18,2
71	3	640 x 480	17,9
71	3	800 x 600	17,7
71	3	1024 x 768	17,6
71	3	1280 x 1024	17,5

Obviamente, reduzir a quantidade de objetos estáticos faz com que a quantidade de quadros por segundo aumente, ou seja, o desempenho será melhor, pois menos polígonos precisarão ser renderizados. A TAB. 5.3 mostra a avaliação de desempenho do ambiente

urbano com diferentes quantidades de objetos estáticos. Nas cinco primeiras linhas, foram desativados os 14 objetos do tipo *Building*, deixando o ambiente com 57 objetos estáticos visíveis. Nas próximas cinco, os treze objetos *Tree* foram desativados, deixando o ambiente com 44 objetos. Continuando, foi desativado o objeto *Stadium Football*, deixando o cenário com 43 objetos. Por fim, foram desativados os cinco objetos do tipo *Mountain House*, deixando o cenário com 38 objetos.

TAB. 5.3 – Avaliação de desempenho com quantidade de objetos variável.

Qtde. Objetos	Qtde. Dirigíveis	Resolução	Quadros/Segundo
57	3	320 x 240	19,0
57	3	640 x 480	18,8
57	3	800 x 600	18,6
57	3	1024 x 768	18,4
57	3	1280 x 1024	18,3
44	3	320 x 240	27,5
44	3	640 x 480	27,1
44	3	800 x 600	27,0
44	3	1024 x 768	26,5
44	3	1280 x 1024	26,1
43	3	320 x 240	27,9
43	3	640 x 480	27,5
43	3	800 x 600	27,2
43	3	1024 x 768	26,7
43	3	1280 x 1024	26,3
38	3	320 x 240	45,6
38	3	640 x 480	44,5
38	3	800 x 600	43,7
38	3	1024 x 768	42,5
38	3	1280 x 1024	42,2

A quantidade de objetos importados influi diretamente no desempenho do simulador. Entretanto, a remoção de objetos de tipos diferentes pode influenciar de forma distinta no desempenho. Isso ocorre porque alguns objetos são mais complexos que outros. A complexidade de um objeto está relacionada principalmente com a quantidade de polígonos e texturas que o compõem. Objetos muito elaborados e detalhados podem ter um forte impacto negativo no desempenho da simulação. Observando a TAB. 5.2 e a TAB. 5.3, é possível perceber que a remoção do objeto *Building* causou pouco impacto no desempenho. Contudo, a remoção dos objetos *Tree* e *Mountain House* teve forte impacto no aumento do desempenho do simulador. Isso ocorre pelo fato de o objeto *Building* ser menos complexo em comparação aos outros dois.

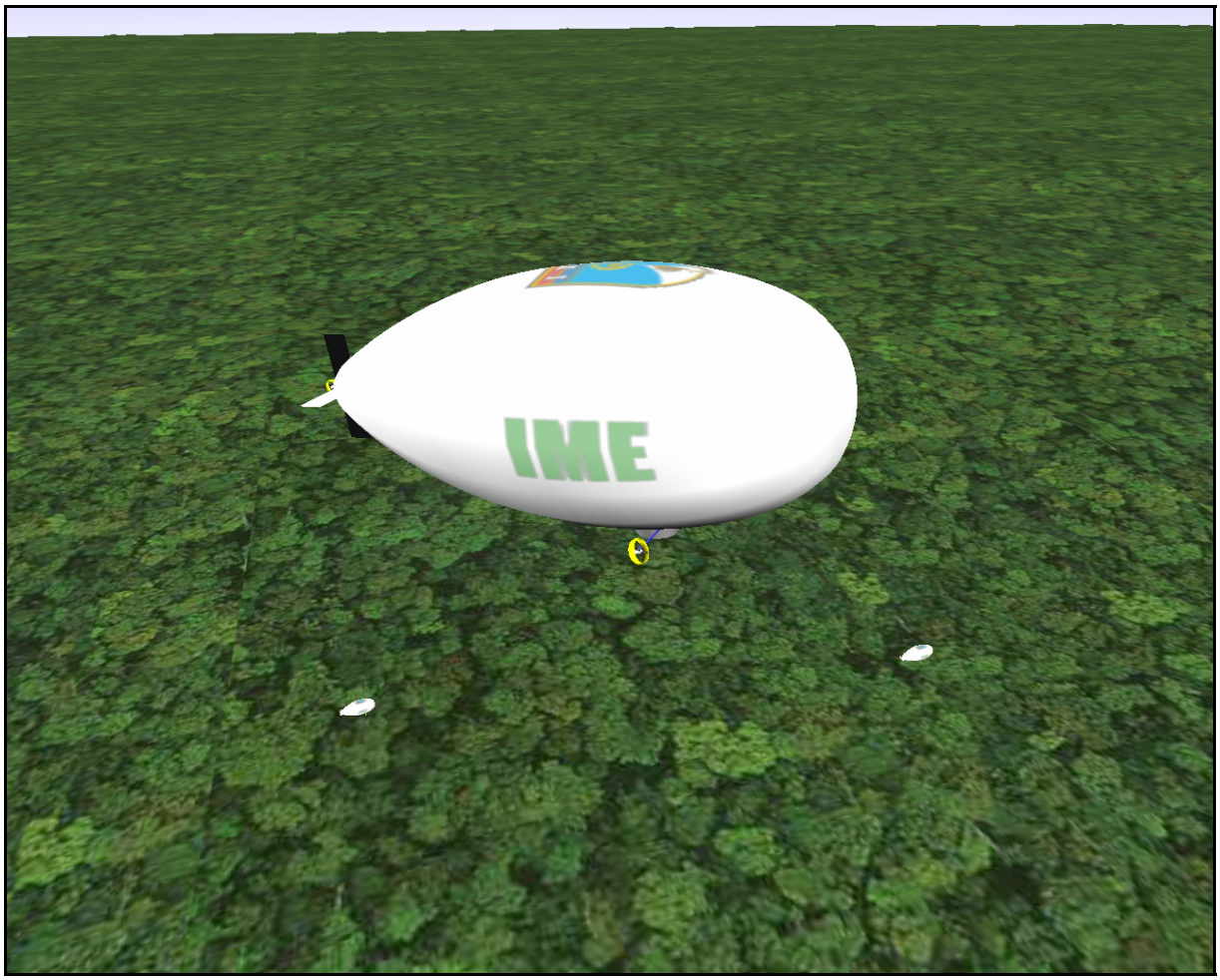

FIG. 5.29 – Visão aérea de um ambiente de floresta sobrevoado por 3 dirigíveis.

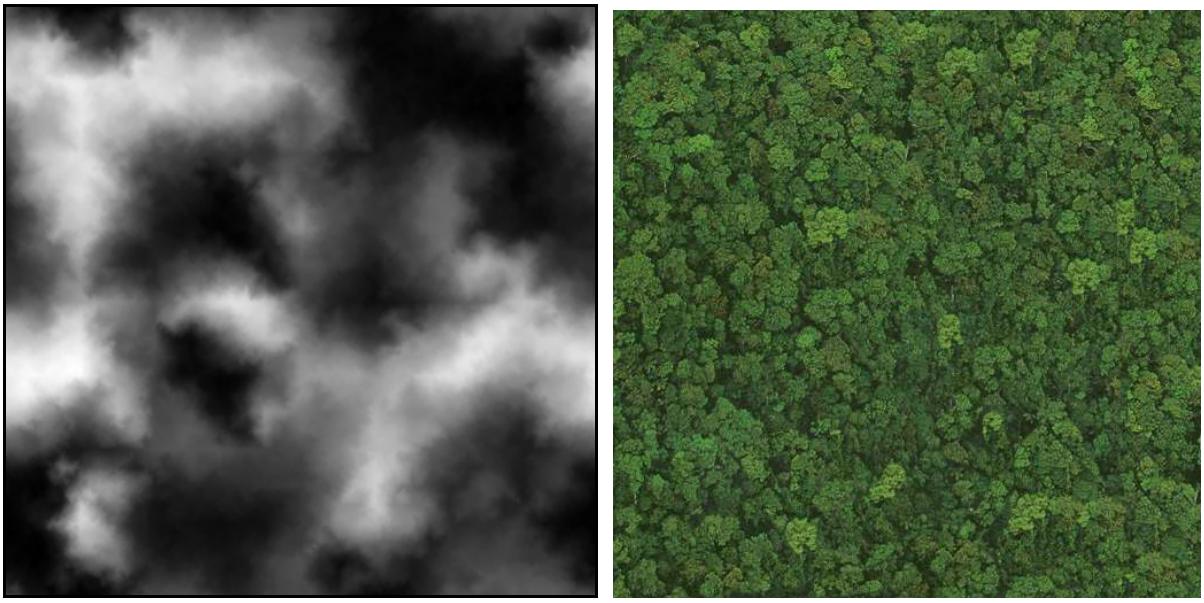

FIG. 5.30 – Mapa de relevo (esquerda) e textura (direita) do ambiente de floresta.

Além de ambientes fechados (*indoors*), é possível criar ambientes abertos (*outdoors*). A FIG. 5.29 mostra uma floresta sendo sobrevoada por uma equipe de três dirigíveis. A floresta foi criada com o mapa de relevo e a textura da FIG. 5.30. O fator de escala utilizado para o mapa de relevo foi de (4, 4; 0,1), respectivamente, para a latitude, longitude e altitude. A TAB. 5.4 apresenta uma avaliação do desempenho do cenário de floresta. O aspecto plano da floresta se deve ao baixo valor do fator de escala para a altitude do mapa de relevo.

TAB. 5.4 – Avaliação do desempenho de renderização do ambiente de floresta.

Qtde. Dirigíveis	Resolução	Quadros/Segundo
3	320 x 240	22,8
3	640 x 480	22,6
3	800 x 600	22,2
3	1024 x 768	22,0
3	1280 x 1024	22,0

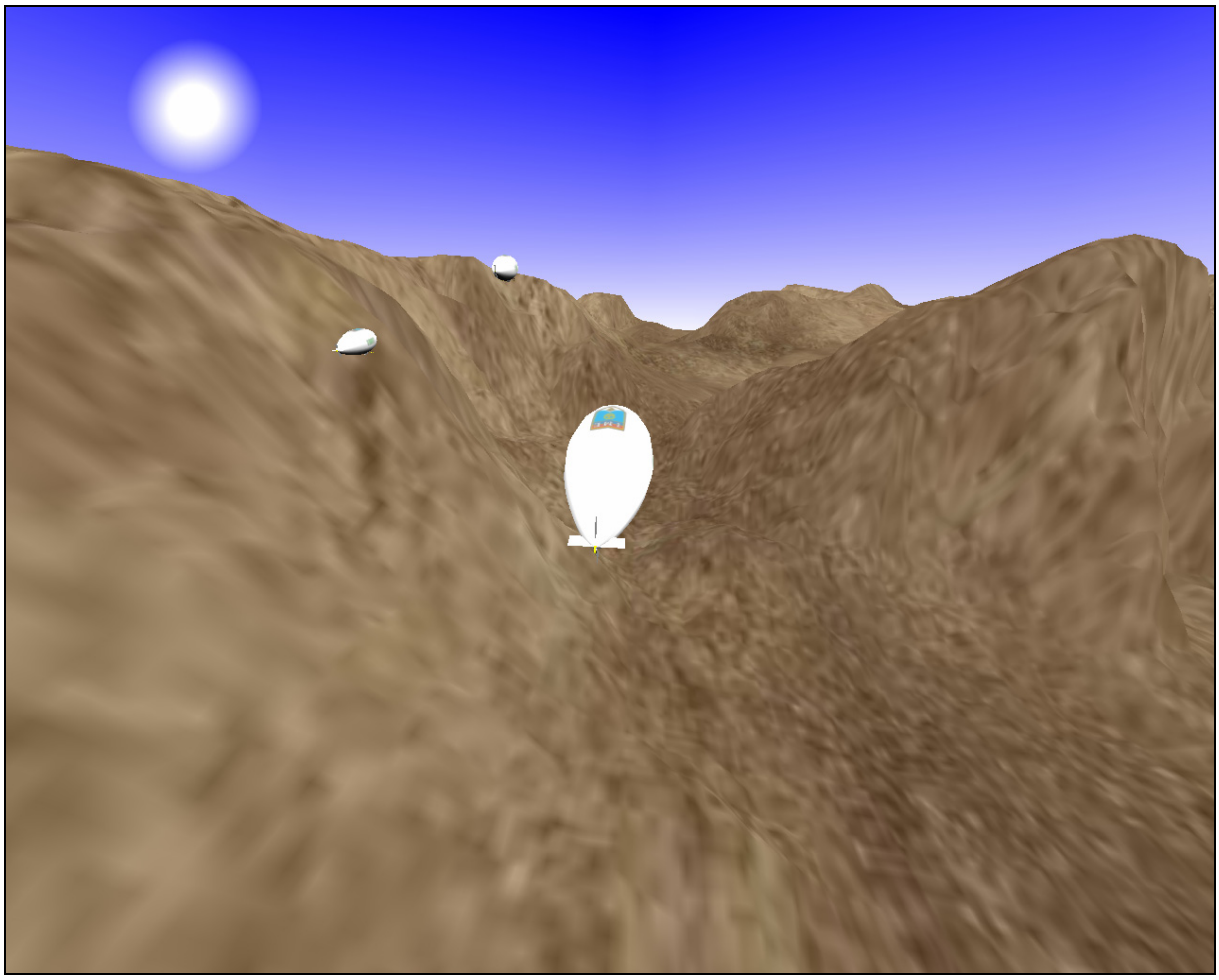

FIG. 5.31 – Região montanhosa desértica, com três dirigíveis pairando.

A FIG. 5.31 mostra uma região montanhosa desértica sendo sobrevoada por uma equipe de três dirigíveis. A região foi criada com o mapa de relevo da FIG. 5.30 e com a textura da FIG. 5.32. O fator de escala utilizado para o mapa de relevo foi de (4; 4; 0,5), para a latitude, longitude e altitude, respectivamente. O aspecto montanhoso se deve ao fator de escala de altitude ser cinco vezes maior do que no ambiente de floresta apresentado anteriormente.

FIG. 5.32 – Textura da região montanhosa.

A TAB. 5.5 mostra a avaliação de desempenho do simulador para a região montanhosa.

TAB. 5.5 – Avaliação do desempenho de renderização do ambiente montanhoso.

Qtde. Dirigíveis	Resolução	Quadros/Segundo
3	320 x 240	21,2
3	640 x 480	20,9
3	800 x 600	20,6
3	1024 x 768	18,0
3	1280 x 1024	17,9

Para concluir a avaliação de desempenho dos gráficos do simulador, a FIG. 5.33 apresenta um ambiente de mar aberto. Uma quantidade variável de dirigíveis é utilizada para verificar o impacto no desempenho. A FIG. 5.34 apresenta o mapa de relevo e a textura utilizada para o ambiente de mar aberto. Na TAB. 5.6, são mostradas as taxas de quadros por segundo para diferentes resoluções e para diferentes quantidades de dirigíveis. De acordo com os dados da TAB. 5.6, é perceptível que a quantidade de dirigíveis presentes no ambiente de

simulação não tem tanto impacto no desempenho do simulador. Isso significa que, graficamente, o dirigível não é um objeto complexo. Cada dirigível presente no simulador requer processamento para calcular o seu comportamento físico, ou seja, sua posição e orientação em resposta às entradas recebidas. O baixo impacto no desempenho gráfico também mostra que o cômputo do modelo matemático não é muito custoso.

TAB. 5.6 – Avaliação de desempenho do ambiente de mar aberto.

Qtde. Dirigíveis	Resolução	Quadros/Segundo
1	320 x 240	22,0
1	640 x 480	20,6
1	800 x 600	20,0
1	1024 x 768	19,0
1	1280 x 1024	18,0
3	320 x 240	21,5
3	640 x 480	20,0
3	800 x 600	19,5
3	1024 x 768	18,5
3	1280 x 1024	18,0
5	320 x 240	21,3
5	640 x 480	20,0
5	800 x 600	19,5
5	1024 x 768	18,5
5	1280 x 1024	18,0
10	320 x 240	20,0
10	640 x 480	19,3
10	800 x 600	19,0
10	1024 x 768	18,5
10	1280 x 1024	18,0

Toda a análise de desempenho da parte gráfica foi feita em um computador com processador Intel Pentium IV de 3.06Ghz, com 2Gb de memória RAM e placa de vídeo ASUS com chipset ATI 9800 XT e 256Mb de memória dedicada, rodando o sistema operacional Microsoft Windows XP Professional SP2. Isso mostra que, mesmo em um computador de configuração modesta para aplicações gráficas, é possível executar o simulador com um bom desempenho. Apesar de o desempenho gráfico ser crucial para a utilização do simulador, o desempenho do mecanismo de acesso remoto também o é. Neste sentido, a subseção seguinte apresenta uma análise de desempenho de uma aplicação cliente acessando o simulador como um servidor de simulação, utilizando comunicação via soquetes, sob diferentes meios de comunicação de dados.

FIG. 5.33 – Ambiente de mar aberto, com três dirigíveis pairando.

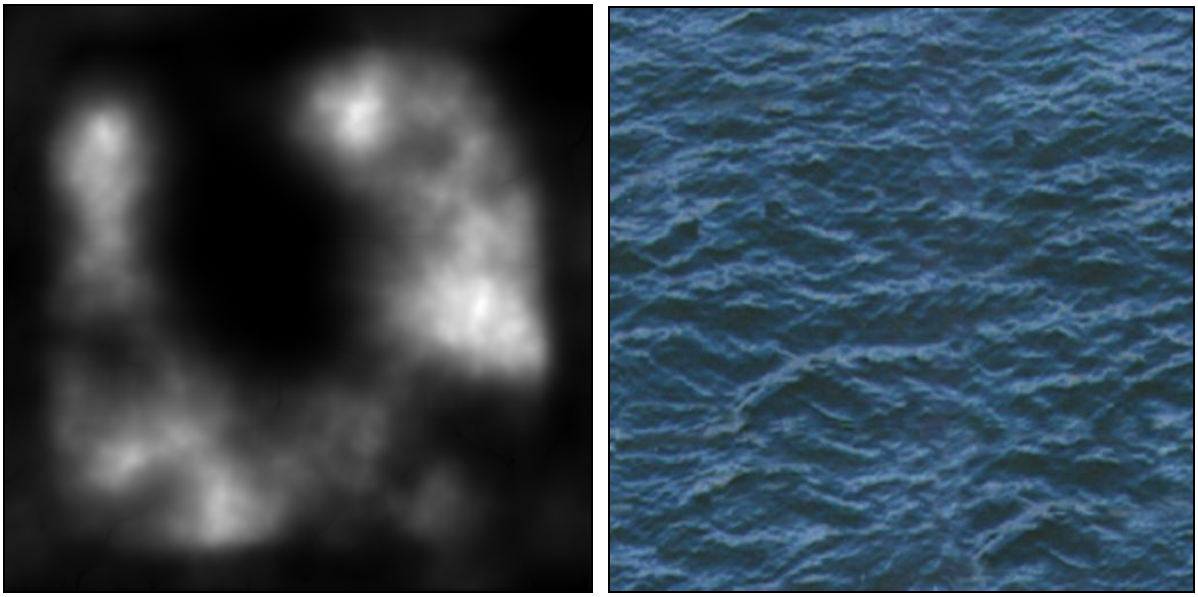

FIG. 5.34 – Mapa de relevo (esquerda) e textura (direita) do ambiente de mar aberto.

5.5.2 AVALIAÇÃO DE DESEMPENHO DA UTILIZAÇÃO REMOTA

Para que o simulador possa ser usado remotamente, além de bom desempenho gráfico ele deve ser capaz de enviar e receber dados com rapidez de/para a aplicação cliente. Esta subseção apresenta alguns resultados de testes realizados com o simulador sendo acessado remotamente por uma aplicação cliente, solicitando e enviando dados. Os testes foram realizados em três diferentes meios físicos: localmente, com cliente e servidor rodando no mesmo computador; usando o padrão *ethernet* de 100 MBits/segundo; e usando o padrão sem-fio 802.11g, de 55MBits/segundo.

O primeiro teste consistiu em solicitar a posição de um dirigível continuamente por um período de 30 segundos. Os resultados dos testes mostraram que esse tipo de consulta, enviando e recebendo algumas poucas dezenas de caracteres, pode ser utilizada sem problemas, independente do meio físico. A TAB. 5.7 apresenta os resultados obtidos.

TAB. 5.7 – Avaliação de desempenho de uma consulta simples ao servidor de simulação.

Tipo de Rede	Respostas/Segundo
Local	3165,567
Ethernet	2755,033
802.11g	723,2

A análise de desempenho de solicitações das imagens das câmeras embarcadas foi um pouco mais elaborada, devido à quantidade de dados que uma imagem contém. Os testes consistiram em solicitar continuamente, durante 30 segundos, imagens de uma das câmeras embarcadas, com o dirigível em movimento. Pelo fato de o canal de comunicação ser um gargalo, para realizar as transferências com maior rapidez, as imagens podem opcionalmente ser convertidas para o formato JPG. Os testes foram feitos utilizando imagens no formato BMP e JPG. O padrão JPG consiste em um formato de imagem compactado. O preço que se paga pelo tamanho menor de uma imagem JPG é o custo computacional. Uma imagem JPG consome processamento para ser gerada e para ser mostrada. O formato BMP, apesar de ser cerca de 80% maior, não precisa de processamento extra para ser gerado ou exibido. Conforme apresentado na TAB. 5.8, ambos os formatos se mostram melhores de acordo com a situação. Os resultados com fundo azul são relativos ao formato JPG e, com fundo branco, ao BMP. Os valores estão em quadros por segundo.

TAB. 5.8 - Avaliação da transmissão de imagens entre cliente x servidor.

Resolução x Meio	320x240	640x480	800x600	1024x768	1280x1024
Local	71,533	13,1	6,833	3,167	1,333
	51,167	13,6	8,7	5,367	3,233
Ethernet	32	6,767	3,9	1,967	0,967
	53,633	13,9	8,9	5,467	3,267
802.11g	6,3	1,8	0,9	0,633	0,367
	36,1	9,4	6,1	3,367	2,533

	BMP
	JPG

Pode-se perceber que, para a resolução de 320 x 240 pixels, quando cliente e servidor estão no mesmo computador, o formato BMP se mostra com melhor taxa de transferência do que o formato JPG. Isso ocorre porque, em vez de o canal de comunicação ser o gargalo, a capacidade de processamento é que passa a sê-lo. Como o formato JPG requer mais processamento, o tempo para a geração das imagens faz a taxa de transferência cair. Um meio de melhorar o desempenho da transferência de imagens seria não transferindo a imagem em si, mas sim os dados necessários para renderizá-la, ou seja, as informações geométricas dos sólidos e as texturas que devem ser aplicadas aos sólidos. Dessa forma, o cliente receberia esses dados e faria a renderização localmente. O aumento de desempenho é conseguido pelo fato de que, em vez de transferir a imagem oriunda do servidor, seriam transferidos apenas os dados para renderizar a imagem na aplicação cliente.

5.6 CONSIDERAÇÕES SOBRE O CAPÍTULO

Esse capítulo teve como foco principal a apresentação de informações sobre a implementação do simulador. A implementação do modelo matemático foi tratada sob um aspecto prático e importantes observações foram feitas. A renderização das imagens também foi abordada de forma mais elucidativa. Foram apresentados detalhes e justificativas sobre a implementação do protocolo de comunicação, que faz do simulador um servidor de simulação capaz de fornecer subsídios para outras aplicações que venham a estender o simulador com algoritmos inteligentes. Além de apresentar maiores detalhes sobre a implementação do simulador, o capítulo apresentou uma avaliação de desempenho que demonstra a viabilidade de utilizá-lo como fonte de subsídios para a implementação de aplicações que visam estender o simulador com algoritmos inteligentes. Foram apresentados testes de desempenho da

renderização dos gráficos tridimensionais e da transferência de dados entre uma aplicação cliente e o servidor de simulação, sob diferentes meios físicos. Os testes mostraram que, mesmo com um computador modesto, é possível tirar proveito de todas as funções do simulador. Ainda como forma de validação, o Capítulo 6 apresenta dois sistemas de determinação de caminhos entre dois pontos, utilizando subsídios fornecidos pelo simulador.

6 DETERMINAÇÃO DE CAMINHOS

O problema de encontrar o caminho mínimo entre dois pontos de um ambiente, de uma rede ou de um grafo é um dos problemas clássicos da Robótica e da Ciência da Computação. Esse problema consiste, normalmente, em encontrar o caminho de menor custo entre dois pontos, considerando a soma dos custos associados aos arcos percorridos.

O problema do caminho mínimo se adapta a diversas situações práticas e, geralmente, é modelado como um grafo. Os nós do grafo seriam os cruzamentos, os arcos seriam as vias, os custos associados aos arcos corresponderiam ao tempo de trajeto ou à distância percorrida, e a solução seria o caminho mais curto (ou mais rápido) entre dois nós do grafo. Particularmente, o interesse maior está na aplicação desse problema na determinação de caminhos para um agente em um ambiente simulado. O agente corresponde a um dirigível robótico e o ambiente corresponde ao espaço de trabalho (ou de atuação) do dirigível.

Neste capítulo, são apresentados dois sistemas de determinação de caminhos que utilizam algoritmos de planejamento e de aprendizado para solucionar o problema do caminho mínimo em sua variante mais simples, ou seja, de um ponto fixo a outro. O problema é atacado utilizando duas abordagens. Primeiramente, é apresentada uma solução para um ambiente tridimensional, utilizando o algoritmo clássico A*, muito utilizado para tal finalidade, principalmente em jogos. Em seguida, é apresentada uma solução usando o algoritmo de aprendizado *Q-Learning*, capaz de aprender em tempo real o caminho entre dois pontos. Vale ressaltar que o objetivo principal do capítulo é demonstrar a possibilidade de extensão do simulador com algoritmos inteligentes, e não, estabelecer algoritmos excepcionais para determinação de caminhos, pois, por si só, a elaboração de algoritmos para tal finalidade é tema de pesquisa que gera diversos trabalhos.

6.1 O AMBIENTE DE DETERMINAÇÃO DE CAMINHOS

O ambiente a ser atacado pelos algoritmos de determinação de caminhos é o espaço de atuação (ou espaço de trabalho) do dirigível. Dois sistemas independentes externos ao simulador foram implementados para encontrar o caminho entre dois pontos desse ambiente, sendo que um utiliza o algoritmo de planejamento A* e outro utiliza o algoritmo de aprendizado *Q-Learning*.

Antes de aplicar os algoritmos mencionados, é necessário discretizar o ambiente, visando reduzir o tamanho do espaço de busca (ou espaço de estados). Nesse sentido, ambos os sistemas utilizam um mecanismo de decomposição em células, transformando o ambiente em um conjunto finito de células adjacentes de dimensões fixas, representado por um vetor de três ou duas dimensões. A obtenção de um caminho contíguo entre dois pontos quaisquer do ambiente pode ser conseguida ligando-se os centros das células que compõem o caminho. A decomposição é realizada pelo sistema, bastando fornecer as dimensões de cada célula.

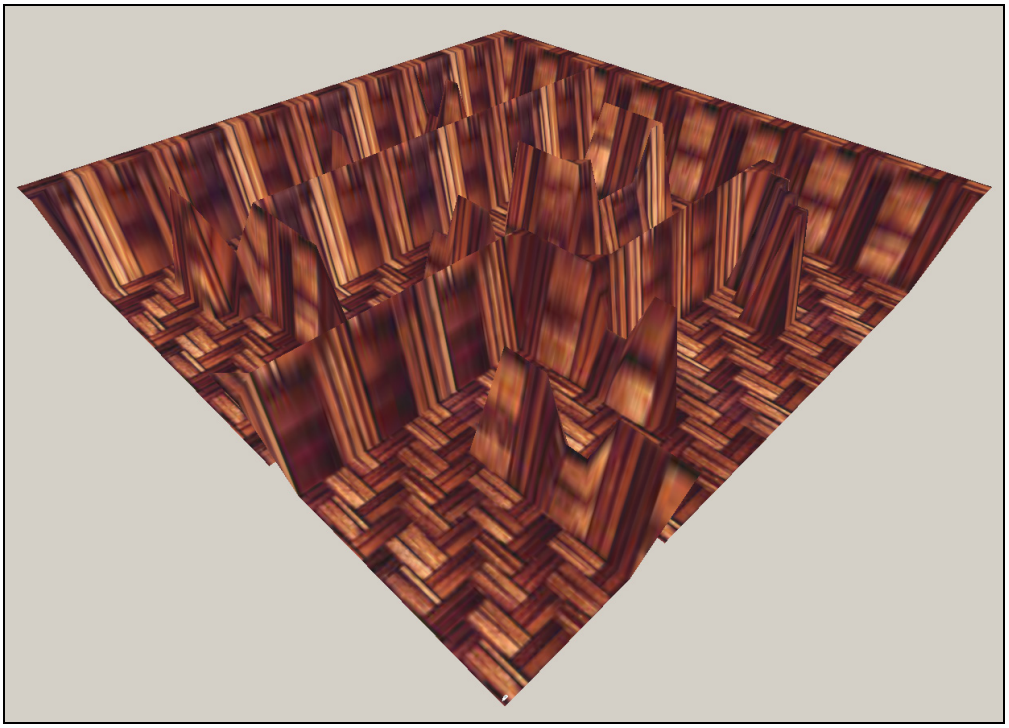

FIG. 6.1 – Ambiente utilizado nos sistemas de determinação de caminhos.

Para a realização dos testes dos sistemas de determinação de caminhos, foi criado um mapa de relevo (*heighfield*) que gera um ambiente um pouco mais desafiador do que o dirigível normalmente encontraria. A FIG. 6.1 apresenta esse ambiente, que consiste em uma seqüência de labirintos pelos quais o dirigível pode se locomover. Algumas paredes possuem fendas, que podem ocasionalmente ser utilizadas como atalho caso seja necessário chegar até o outro lado da parede. A FIG. 6.2 mostra o mapa de relevo utilizado para a geração desse ambiente. Comparando o mapa de relevo da FIG. 6.2 com o ambiente 3D da FIG. 6.1, é possível verificar que as linhas brancas correspondem às paredes, os pequenos traços em cinza correspondem às fendas nas paredes e a parte preta corresponde ao chão.

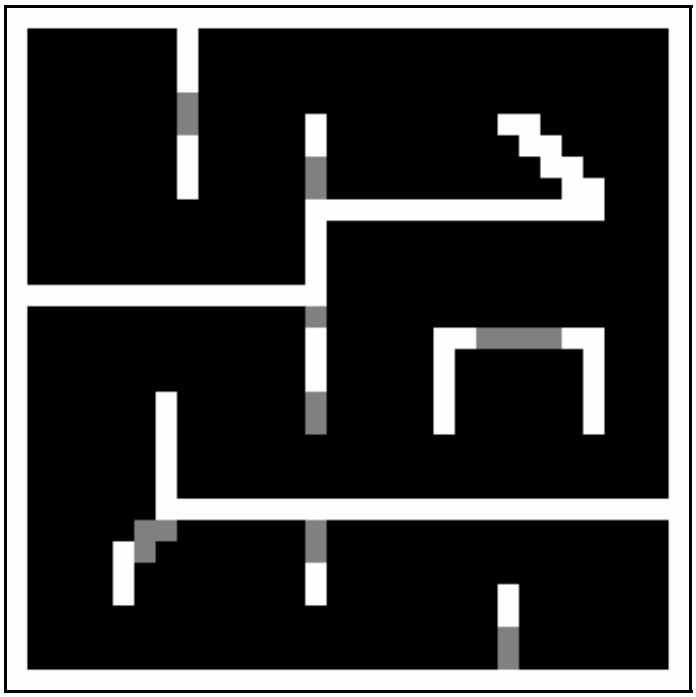

FIG. 6.2 – Mapa de relevo utilizado no ambiente da FIG. 6.1.

O mapa de relevo, que na FIG. 6.2 aparece bem ampliado, corresponde a uma imagem de 32 x 32 pixels. Apesar de ser aparentemente pequeno, se a decomposição em células considerar que cada pixel equivale a uma célula, um espaço de busca de 1.024 células (ou estados) é obtido, sendo que, com exceção das células que correspondem a obstáculos, todas as células podem ser ocupadas pelo dirigível. Ainda, quando se passa a considerar a altura, representada pelo valor de um dos componentes RGB de cor do pixel, deve-se multiplicar o tamanho do espaço de busca por 256, resultando em 262.144 células. Entretanto, para reduzir o espaço de buscas, pode-se configurar as dimensões da célula para que ela abranja mais pixels. Como exemplo, se no caso anterior as dimensões da célula fossem todas iguais a 4 pixels, o espaço de busca seria reduzido para 2.048 estados. Considerando que as dimensões da célula são configuráveis, pode-se obter o tamanho do espaço de estados através da equação

$$T(S) = (mw/cw) * (ml/cl) * (256/ch), \qquad \text{(EQ 6.1)}$$

onde $T(S)$ é o tamanho do espaço de estados, mw e ml correspondem, respectivamente, ao valor em pixels da largura e do comprimento da imagem do mapa de relevo e cw, cl e ch são, respectivamente, a largura, o comprimento e a altura da célula. O valor constante 256 corresponde ao valor máximo de um componente de cor do pixel, que varia de 0 a 255.

Um cuidado extra a ser tomado é que, muitas vezes, uma simples configuração no tamanho da célula possibilita a criação de espaços de busca anteriormente inviáveis de serem atacados por algoritmos de determinação de caminhos. Para ambientes muito grandes, pode-se utilizar células com dimensões maiores, com o cuidado de não exagerar e distorcer demais o formato real do ambiente. Nesse sentido, uma observação importante a ser feita ao se determinar as dimensões da célula diz respeito à inclusão de obstruções inexistentes no ambiente real. Por exemplo, se uma célula tem todas as dimensões iguais a 8, mesmo que um único pixel da região da célula tenha altura superior à zero, a célula será considerada um obstáculo com a altura do pixel de maior valor. Isso pode incorrer na obstrução de um caminho que seja factível no ambiente real.

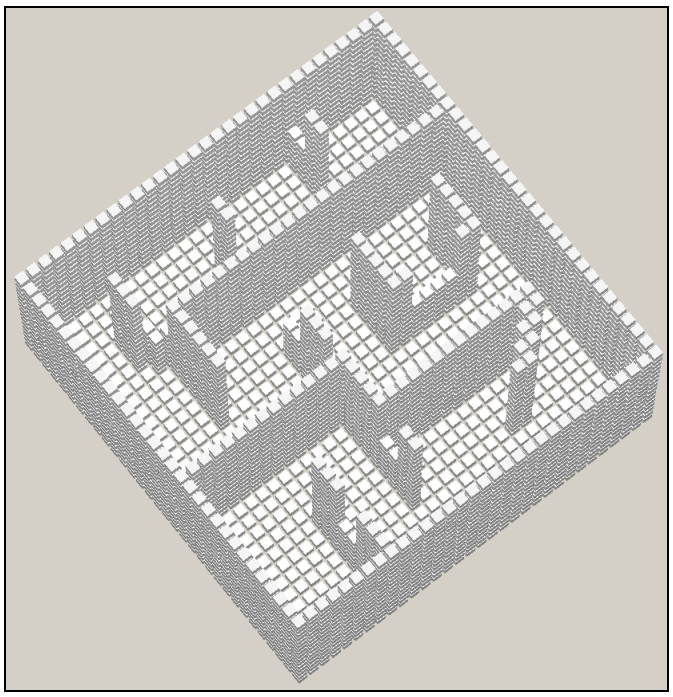

FIG. 6.3 – Decomposição tridimensional em células do ambiente da FIG. 6.1.

A FIG. 6.3 apresenta uma decomposição em células em três dimensões e a FIG. 6.4, em duas dimensões, ambas do ambiente mostrado na FIG. 6.1. A decomposição em duas dimensões é obtida fixando-se a altitude de navegação do dirigível. Dessa forma, as células são configuradas como obstáculo apenas se o relevo do terreno impedir a navegação na altitude fixada. Ao observar a FIG. 6.3 e a FIG. 6.4, é possível notar que as fendas nos obstáculos da FIG. 6.3 são consideradas células livres na FIG. 6.4. Isso ocorre porque a

altitude fixada para obter a decomposição bidimensional é superior à altura onde as fendas iniciam.

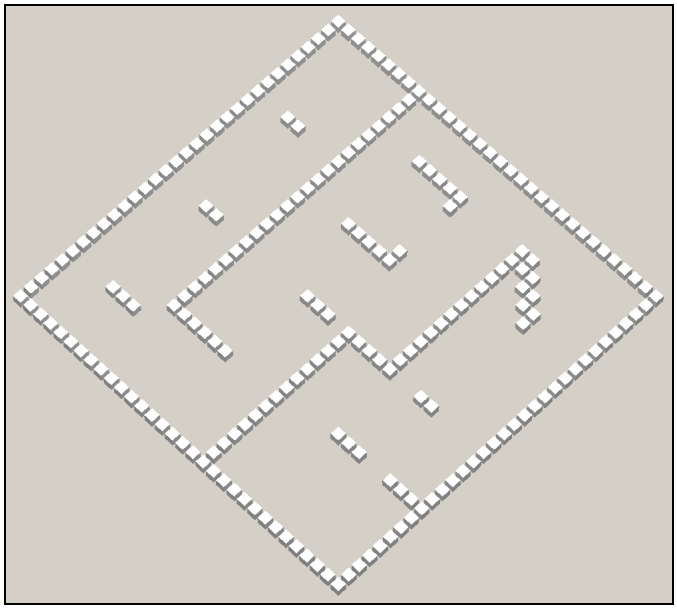

FIG. 6.4 – Decomposição bidimensional em células do ambiente da FIG. 6.1.

A abordagem usando a decomposição tridimensional é utilizada no sistema que ataca o problema de determinar um caminho entre dois pontos utilizando planejamento, que, nesse caso, utiliza o algoritmo A*. O algoritmo *Q-Learning* também poderia ser utilizado. Entretanto, como o tamanho do espaço de buscas na decomposição tridimensional aumenta consideravelmente com relação ao tamanho do espaço de buscas na decomposição bidimensional, a convergência do algoritmo fica extremamente lenta, não sendo possível determinar um caminho em um tempo aceitável para o sistema de dirigíveis em questão.

A decomposição em três dimensões aumenta não só a quantidade de estados no espaço de busca, mas também a quantidade de movimentos (ações) possíveis de serem executados pelo agente. Em um espaço bidimensional, como o agente se movimenta em um plano, apenas 8 movimentos são possíveis: frente, trás, direita, esquerda e para as 4 diagonais. Em um espaço tridimensional, são 26 movimentos possíveis. O sistema que implementa o *Q-Learning* utiliza o espaço de buscas bidimensional. As duas abordagens são discutidas mais detalhadamente nas subseções seguintes. Outra forma de discretizar o ambiente é através da decomposição em células convexas (LATOMBE, 1991), e pode ser utilizada em futuros sistemas inteligentes que visam interagir com o simulador, pois pode apresentar vantagens em algumas situações.

6.2 DETERMINAÇÃO DE CAMINHOS USANDO PLANEJAMENTO

A resolução do problema de determinação de caminhos utilizando algoritmos clássicos de planejamento requer o conhecimento de todo o espaço de busca a ser explorado. Com esse conhecimento, é possível aplicar diversos algoritmos. Dentre os principais algoritmos de busca está o algoritmo de Dijkstra (DIJKSTRA, 1959), que resolve o problema com um único ponto de partida em grafos cujas arestas tenham peso maior ou igual a zero. Sem reduzir o desempenho, esse algoritmo é capaz de determinar o caminho mínimo partindo de um vértice inicial e chegando a todos os outros vértices do grafo. Outro algoritmo é o de Bellman-Ford (BELLMAN, 1958), que resolve o problema para grafos com um único ponto de partida e com arestas que podem ter custos negativos. O algoritmo de Floyd-Warshall (FLOYD, 1962) é capaz de determinar a distância entre todos os pares de vértices de um grafo. Finalmente, tem-se o algoritmo A* (HART *et al.*, 1968), que é um algoritmo heurístico capaz de calcular o caminho mínimo a partir de um único vértice fonte. A diferença básica entre o algoritmo A* e o de Dijkstra é que o primeiro usa uma componente heurística para tentar melhorar a escolha do próximo vértice (ou estado) a ser visitado, visando acelerar o processo. A próxima subseção apresenta detalhes do sistema de determinação de caminhos que utiliza o algoritmo A*.

6.2.1 USANDO O ALGORITMO A* (A-ESTRELA)

O sistema de determinação de caminhos usando o algoritmo A* encontra um caminho entre um único estado inicial e um único estado final. Antes de executar o algoritmo em si, é preciso obter o mapa de relevo do ambiente do servidor de simulação, enviando o comando AMBHTMAP, conforme o protocolo descrito na subseção 5.5.1. Após obter o mapa de relevo, é realizada uma decomposição em células, gerando um conjunto de células idêntico ao da FIG. 6.3. Finalmente, determina-se a posição do objetivo e o índice do dirigível cuja posição é tomada como ponto de partida. Ao iniciar a execução do algoritmo, a posição do dirigível é requisitada ao servidor de simulação usando o comando BLMPOSIT seguido do índice do dirigível.

A FIG. 6.5 mostra a tela do sistema em execução. Nessa figura, a pequena esfera dourada na parte superior é o objetivo e o cubo azul é o agente. Um fato importante a ser mencionado é que o algoritmo A* nem sempre é capaz de retornar uma solução ótima. Para determinados

ambientes, ele retorna um caminho factível, mas não ótimo. Na maioria dos casos, a solução obtida é próxima da ótima.

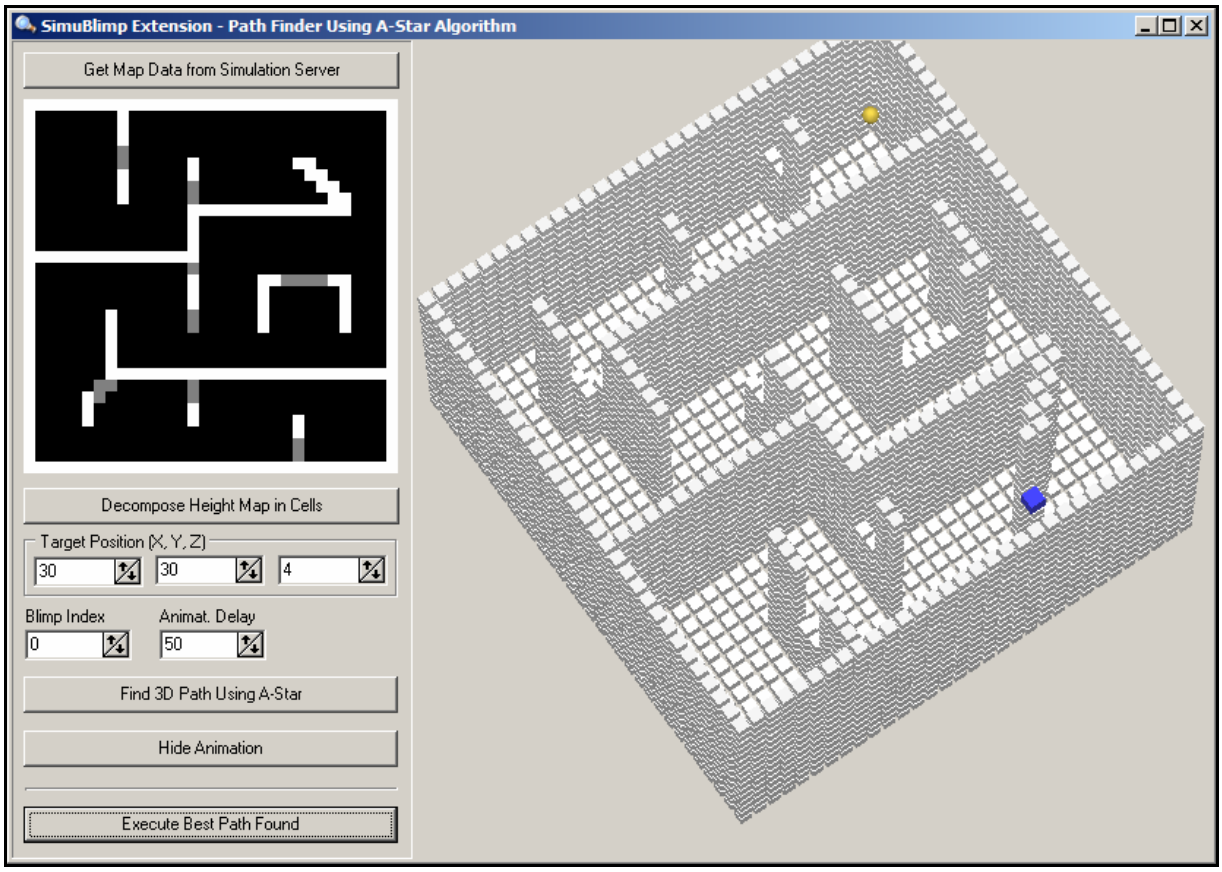

FIG. 6.5 – Sistema de determinação de caminhos com o algoritmo A*.

No ambiente decomposto em células da FIG. 6.5, que corresponde ao ambiente da FIG. 6.1, o algoritmo não foi capaz de retornar o caminho ótimo, ou, pelo menos, próximo do ótimo, possivelmente devido à complexidade do ambiente e/ou ao método utilizado para determinar a parte heurística, responsável por "guiar" o processo de busca.

Para não prejudicar um dos principais benefícios do A*, que é o seu desempenho, o valor da parte heurística é obtido de forma simples e corresponde ao maior valor dentre os valores das coordenadas X, Y e Z do estado em questão, considerando o objetivo como a origem do sistema de coordenadas, ou seja, na posição (0, 0, 0). Ainda com relação ao desempenho, o algoritmo foi capaz de determinar um caminho até o objetivo usando 181 passos e 78 mudanças de direção, o que significa que 181 células compõem o caminho e que, ao percorrê-lo, o dirigível terá que rotacionar um determinado ângulo por 78 vezes. A observação da

quantidade de mudanças de direção é importante, pois uma mudança de direção requer tempo e energia extras.

Ao contrário do algoritmo A*, o algoritmo apresentado na subseção 6.4, apesar de ter algumas restrições, é capaz de encontrar um caminho mais próximo do ótimo na maioria das vezes, sendo que, a cada execução, o caminho encontrado pode ser ligeiramente diferente, como é mostrado mais adiante.

Após a determinação do caminho, é possível enviar as coordenadas das células componentes do caminho para o simulador, de forma a criar uma trajetória a ser seguida pelo dirigível cuja posição foi tomada como posição inicial do agente. Para informar ao servidor de simulação as coordenadas de um novo nó de uma trajetória a ser seguida por um dirigível, deve-se utilizar o comando BPPATHND, concatenado com o índice do dirigível, e seguido dos valores das componentes X, Y e Z da coordenada do ponto, da velocidade a ser utilizada (em m/s) e do ângulo a ser rotacionado (em graus) para ir até o próximo nó, com uma quebra de linha entre cada valor. Por exemplo, se a célula de índice (32, 6, 8) estiver presente na trajetória determinada, ela pode ser enviada para o compor a trajetória do dirigível de índice 0 através do comando: "BPPATHND + QL + 0 + QL + 32 + QL + 6 + QL + 8 + QL + 4 + QL + 33". Esse comando instrui o dirigível a utilizar a velocidade de 4 m/s e rotacionar 33 graus até o próximo nó. Após passar todos os nós da trajetória, basta enviar o comando BPPATHFW seguido do índice do dirigível para que a trajetória seja executada por esse dirigível.

6.3 DETERMINAÇÃO DE CAMINHOS USANDO APRENDIZADO POR REFORÇO

Conforme apresentado na subseção anterior, apesar de ser bastante popular quando se trata de algoritmos de determinação de caminhos, o algoritmo A* nem sempre retorna um caminho bom. Um dos fatores que contribui para isso é o fato de a componente heurística limitar a quantidade de estados a ser visitada, impedindo uma exploração maior do ambiente. Certamente, uma forma mais elaborada de determinar o valor da componente heurística pode melhorar o caminho encontrado, mas, conseqüentemente, pode reduzir o desempenho do algoritmo.

De acordo com os resultados aqui obtidos, a utilização de algoritmos de aprendizado (como o *Q-Learning*) para resolução do problema de determinação de caminhos, muitas vezes

pode levar a resultados mais satisfatórios. Quando bem parametrizado e implementado, o algoritmo *Q-Learning* permite a inclusão de diversas restrições para compor a solução do problema. Sendo o algoritmo baseado em um mecanismo de aprendizado através de punição e recompensa, é possível lançar mão desse mecanismo para valorizar algum comportamento específico por parte do agente. A seguir, são apresentados detalhes do sistema de determinação de caminhos que utiliza o algoritmo *Q-Learning* para encontrar o caminho entre dois pontos.

6.3.1 USANDO O ALGORITMO Q-LEARNING

Um dos principais diferenciais da implementação utilizando AR para a anterior (que usou o A*) está na possibilidade de inclusão de algumas restrições extras para compor a solução do problema. Isso quer dizer que, além de aprender um caminho cuja distância é próxima da ótima, é possível aprender um comportamento para seguir esse caminho. Além disso, a obrigação de ter um modelo do ambiente deixa de existir, e o forte caráter exploratório variável do agente possibilita a descoberta de caminhos bem inusitados.

Ao percorrer um caminho qualquer, intuitivamente é possível notar que, quando for possível andar em linha reta, deve-se sempre fazê-lo, pois, como não é segredo, o caminho mais curto entre dois pontos é uma reta. Em um sistema que usa AR para a determinação de caminhos, isso significa que qualquer mudança de direção desnecessária por parte do agente deve ser punida, e assim acontece no sistema aqui apresentado. A FIG. 6.6 mostra a tela do sistema.

O sistema é bem semelhante ao anterior, diferindo no fato de se utilizar a decomposição bidimensional em células (explicada anteriormente) e o algoritmo *Q-Learning* em vez do A*. Apesar de parecer uma limitação grave, o fato de o caminho computado ser respectivo à navegação do dirigível em uma altitude fixa, isso não quer dizer que o dirigível deverá navegar numa altitude exata. Na realidade, o cômputo do caminho é feito para uma faixa de altitude, que por sua vez é determinada pela altura da célula, informada antes de se realizar a decomposição em células. Essa característica dá ao dirigível uma margem de espaço para algumas inevitáveis variações de altitude que podem ser provocadas por manobras e por perturbações externas.

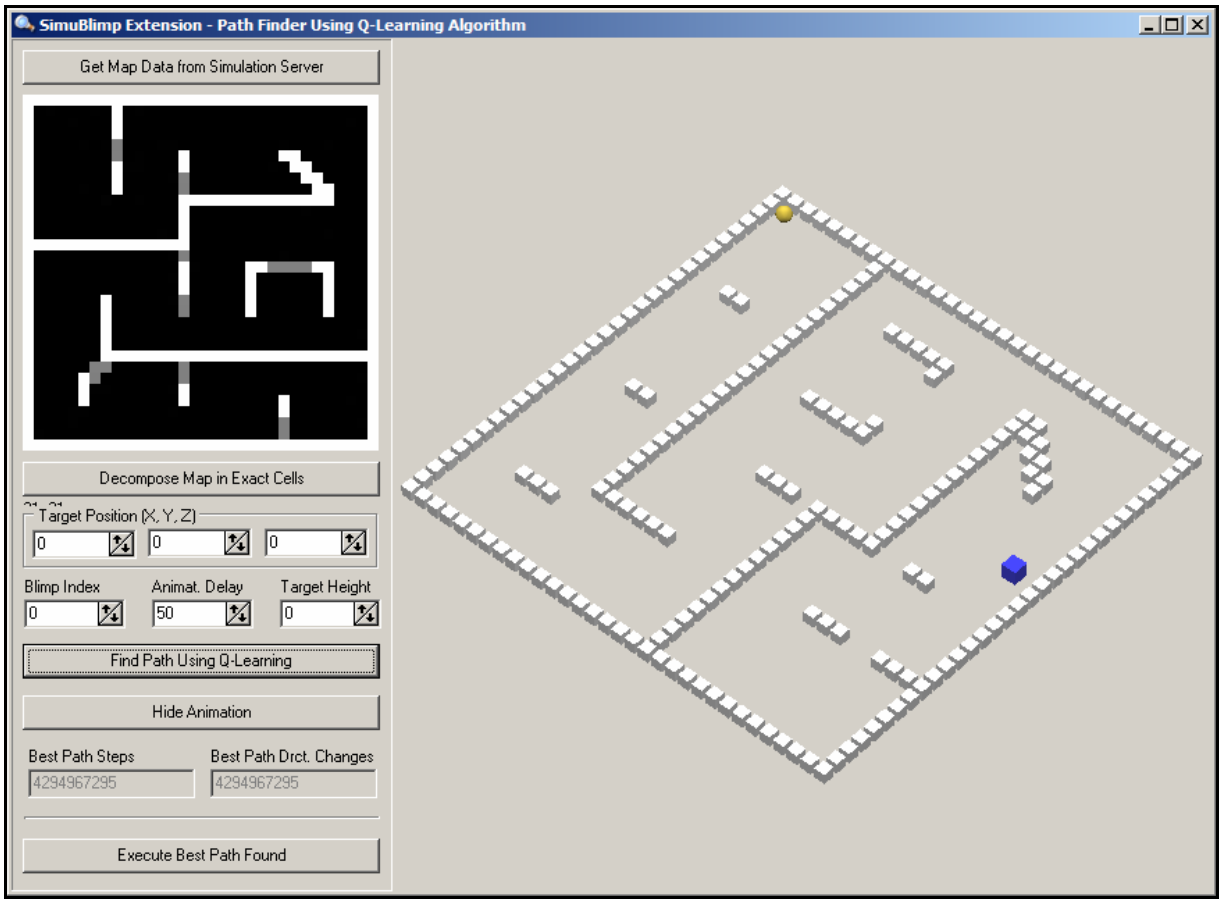

FIG. 6.6 – Sistema de determinação de caminhos com o algoritmo *Q-Learning*.

A execução do algoritmo *Q-Learning* para o mesmo ambiente da FIG. 6.1 resultou em um caminho muito melhor do que o obtido pelo algoritmo A*. Como o algoritmo *Q-Learning* pode ter resultados ligeiramente diferentes a cada execução, a TAB. 6.1 apresenta os resultados de uma série de 100 execuções para avaliar o desempenho do algoritmo.

TAB. 6.1 – Execução do *Q-Learning* sobre o ambiente da FIG. 6.1.

Células	Mudanças de Direção	Ocorrência (%)
80	14	2
80	17	1
80	19	1
79	11	2
79	12	6
79	13	5
79	14	12
79	15	18
79	16	17
79	17	13
79	18	10
79	19	7
79	20	6

Como é mostrado na TAB. 6.1, todas as execuções resultaram em caminhos melhores do que o obtido com o algoritmo A*. A tabela mostra que, em apenas 2% das execuções, o algoritmo obteve o caminho ótimo (com 79 células e 11 mudanças de direção) mas, na maioria das vezes, ou seja, em 18% dos casos, o algoritmo alcançou um resultado próximo do ótimo, com diferença apenas na quantidade de mudanças de direção, que ocorreram 15 vezes contra 11 do caminho ótimo.

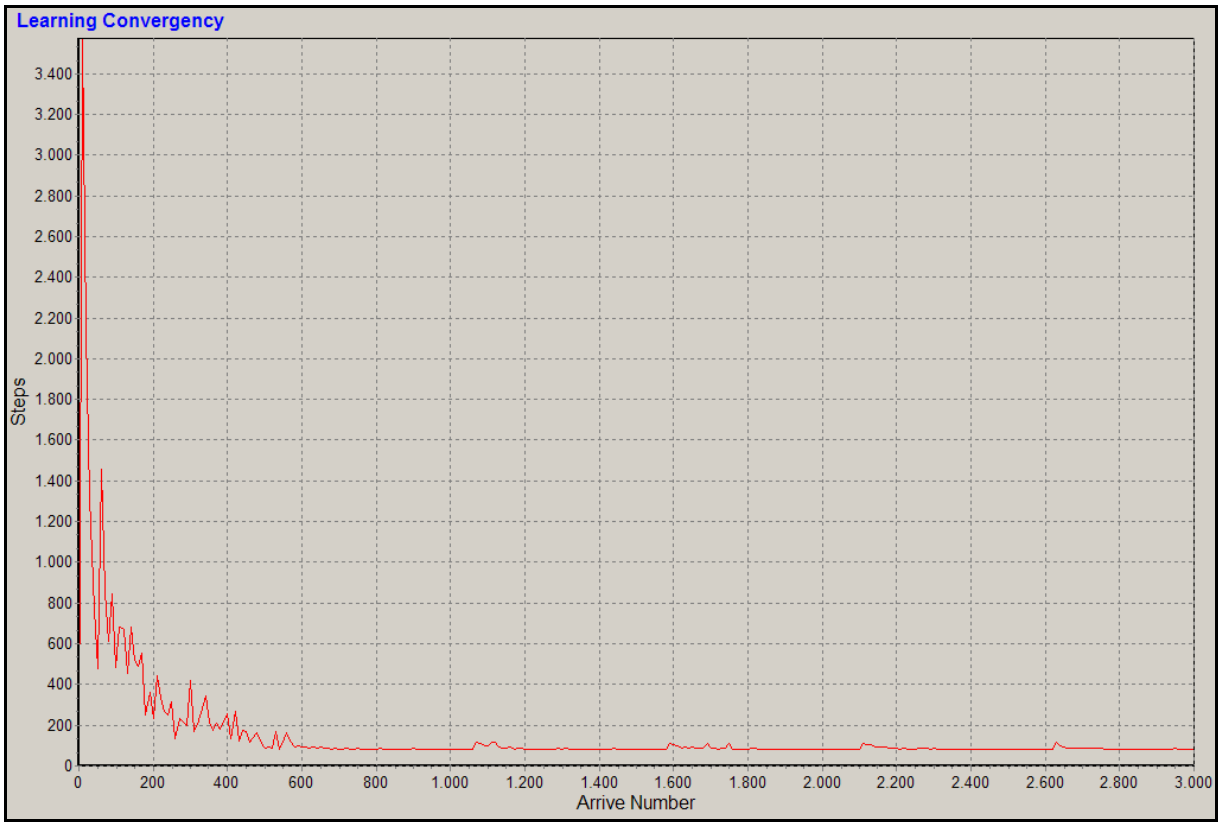

FIG. 6.7 – Gráfico de convergência do algoritmo *Q-Learning*.

O gráfico da FIG. 6.7 apresenta o comportamento do algoritmo *Q-Learning* em 3.000 iterações, sendo que cada iteração equivale a sair do ponto de partida e alcançar o objetivo. Através do gráfico, pode-se ver que o algoritmo converge e que o agente atinge o objetivo, pela primeira vez, após, aproximadamente, 3.400 passos (eixo *Steps*). Daí em diante, cada nova chegada ao objetivo geralmente leva menos passos. Mesmo após obter o caminho de menor custo, fato que acontece em torno da iteração de número 700, o agente não perde o caráter exploratório, e esporadicamente tenta um caminho diferente do já computado. Isso pode ser observado através dos pequenos saltos no gráfico em torno das iterações 1.100,

1.600, 1.700 etc., em que a quantidade de passos fica um pouco acima da menor já encontrada, que foi 79.

Os resultados apresentados foram obtidos utilizando uma taxa de aprendizado igual a 0,9 e um fator de desconto de 0,99. O caráter exploratório do agente é obtido selecionando uma ação aleatória 25% das vezes em que o agente deve executar um movimento. Para acelerar a convergência, a cada 10 chegadas ao objetivo esse percentual é reduzido, visando aproveitar as melhores ações já encontradas até o momento, ou seja, explotando. A redução do caráter exploratório é feita de forma que, mesmo após obter a menor quantidade de passos, pelo menos 2% das ações executadas no ambiente com um caminho já encontrado serão aleatórias. Foi estabelecido um critério de parada igual a 3.000 iterações, pois, nos testes realizados, o caminho raramente foi melhorado após essa quantidade de iterações.

Como o algoritmo tem o objetivo de minimizar o custo do caminho obtido, o agente deve receber a menor quantidade de recompensas possíveis. Na realidade, quanto mais positivo for o sinal numérico dado ao agente em resposta a uma ação, pior é essa ação, pois o sinal numérico corresponde ao custo de transição entre os estados. Desde que usado apropriadamente, não faz diferença utilizar recompensas positivas ou negativas.

Uma comparação direta entre o A* e o *Q-Learning*, da forma como eles foram implementados neste trabalho, seria injusta, tendo em vista que o espaço de buscas atacado pelo algoritmo A* é bem superior ao usado com o *Q-Learning*. O primeiro considera todo o espaço tridimensional e o segundo considera apenas uma "fatia horizontal" desse espaço. A utilização dessa técnica, apesar de desconsiderar boa parte do ambiente, é plausível, tendo em vista que a navegação em uma faixa de altitude limitada é algo totalmente viável. Para ilustrar melhor a aplicabilidade do *Q-Learning*, cuja utilização não é tão popular em problemas de determinação de caminhos, a FIG. 6.8 apresenta quatro ambientes decompostos em células e os resultados obtidos atacando-os com o algoritmo *Q-Learning*. Os caminhos obtidos são ótimos ou estão bem próximos do ótimo. Na seqüência, da esquerda para a direita, de cima para baixo, a "quantidade de passos"/"mudanças de direção" dos caminhos foram 32/4, 57/7, 47/4 e 47/6, sendo que todos os caminhos foram computados em menos de 5 segundos em um computador de configuração idêntica ao citado no final da subseção 5.6.1.

6.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou dois sistemas de determinação de caminhos que utilizam algoritmos distintos para resolver o problema de determinar o caminho entre dois pontos fixos em um ambiente, demonstrando como estender o simulador com algoritmos inteligentes. Apesar de o objetivo principal do capítulo não ter sido o de desenvolver algoritmos excepcionais para a determinação de caminhos entre dois pontos, as subseções anteriores também mostraram que técnicas que não envolvem planejamento também podem ser utilizadas com eficiência para atacar tal problema. Em especial, o algoritmo de Aprendizado por Reforço *Q-Learning* apresentou algumas características interessantes e seu uso deve ser levado em consideração. Dentre as principais vantagens apresentadas pelo algoritmo, estão a sua capacidade de ser utilizado em ambientes desestruturados ou semi-estruturados e a sua facilidade em permitir a inserção de restrições através da manipulação do mecanismo de punição e recompensa.

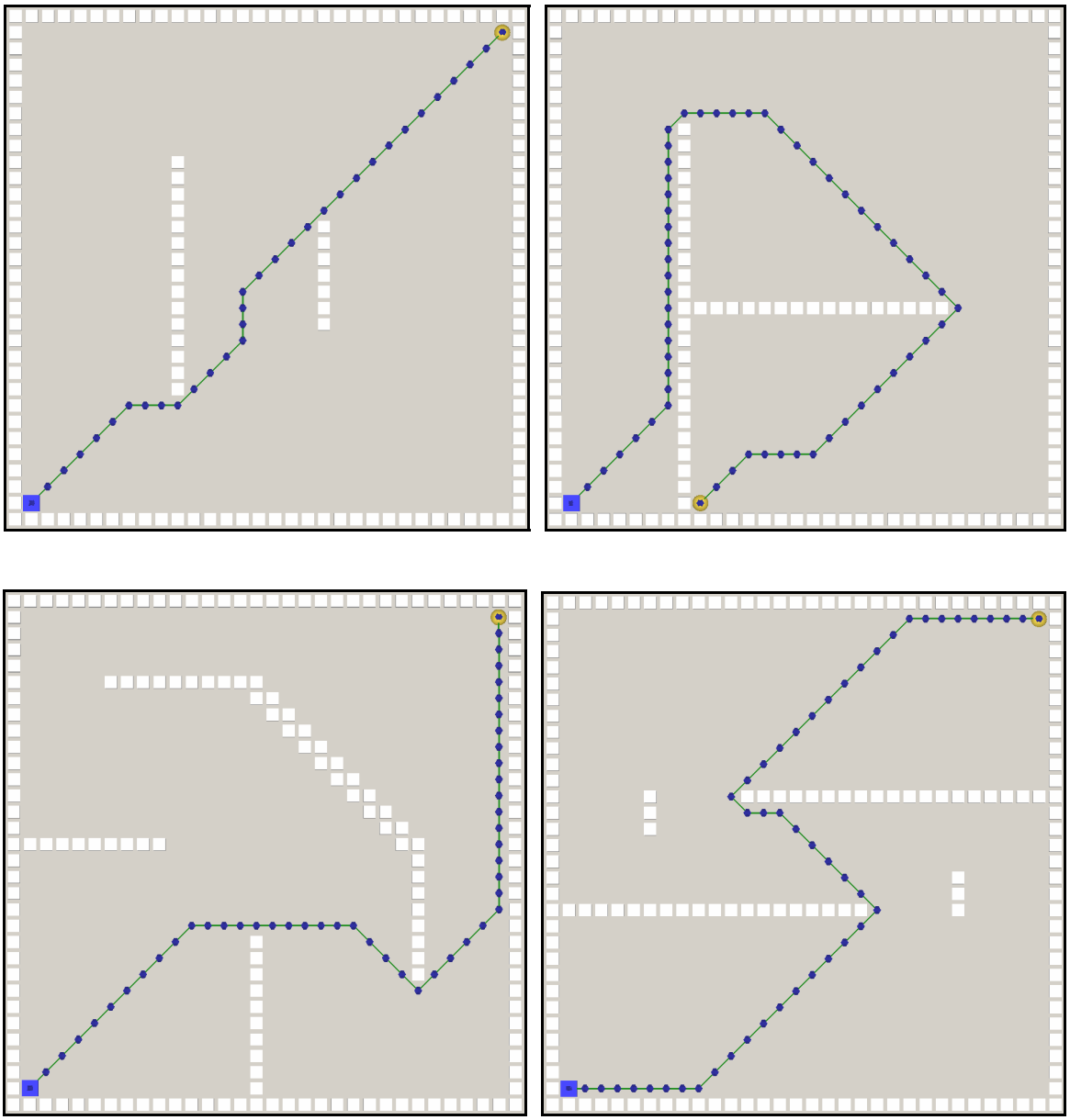

FIG. 6.8 – Ambientes com caminhos determinados pelo *Q-Learning*.

Os sistemas de determinação de caminhos implementados são capazes de interagir com o servidor de simulação, sendo que essa interação se inicia com a obtenção do mapa de relevo do ambiente simulado. Posteriormente, o mapa é discretizado, através do método de decomposição em células, para reduzir o tamanho do espaço de buscas. Após a discretização do ambiente, os algoritmos são aplicados para encontrar o caminho entre a posição de um dos dirigíveis virtuais do servidor de simulação e um ponto objetivo qualquer pertencente ao ambiente. Computado o caminho, os sistemas podem enviá-lo para criar uma trajetória a ser seguida pelo dirigível que foi usado na obtenção do ponto de partida.

É possível observar, na FIG. 6.8, que o algoritmo buscou minimizar não só a distância percorrida pelo agente (neste caso, o dirigível), mas também a quantidade de mudanças de direção executadas pelo agente. A inserção de algumas restrições, punindo o agente quando ele muda de direção, permitiu que as trajetórias geradas fossem mais suaves. O algoritmo foi parametrizado de forma que a menor distância sempre prevalecesse, e, complementarmente, as trajetórias de distância mínima e com a quantidade mínima de mudanças de direção.

No capítulo seguinte são apresentados alguns comentários gerais que concluem a presente dissertação. São apresentadas também algumas sugestões para trabalhos futuros, visando dar continuidade do trabalho iniciado por (PINHEIRO, 2006) e, posteriormente, continuado por (VIDAL, 2007) e pela presente dissertação, dentro do contexto do projeto VANT do IME.

7 CONSIDERAÇÕES FINAIS

O desenvolvimento de uma equipe de dirigíveis robóticos autônomos capazes de realizar, principalmente, atividades de vigilância e monitoramento, tanto para ações civis como militares, está longe de ser uma tarefa simples. Devido à quantidade de subproblemas inerentes ao problema geral, certamente este trabalho e os já desenvolvidos anteriormente compõem apenas os passos iniciais para alcançar o objetivo maior almejado pelo projeto VANT do IME. Além da quantidade, os subproblemas inerentes são extremamente multidisciplinares, o que demanda a participação de pessoas das mais diversas áreas da Engenharia.

Mesmo se tratando de um problema complexo, a evolução tecnológica dos equipamentos necessários para construção de aeronaves e de seus sistemas intrínsecos tem sido um fator favorável, principalmente porque os equipamentos têm ficado mais leves, mais precisos e menores. Mas, o grande problema passa a ser o custo. Com os sensores existentes na atualidade, a construção do veículo e de sistemas de controle, navegação, reconhecimento etc., se tornam menos complexa, e dependem quase que exclusivamente do uso correto dos dados oriundos desses sensores, que, principalmente quando miniaturizados, tem um custo elevado. O desenvolvimento de uma equipe de dirigíveis robóticos autônomos de baixo custo requer a utilização de sensores de baixo custo, que, por sua vez, requerem o desenvolvimento de sistemas que utilizem esses sensores como principal fonte de dados. Além disso, a simulação é uma técnica muito utilizada para reduzir custos de desenvolvimento e teste desse tipo de aplicação, sendo que o foco principal do presente trabalho foi o desenvolvimento de um simulador para a aplicação em questão, cujas conclusões são apresentadas na próximo subseção.

7.1 CONCLUSÕES

O presente trabalho, somado aos esforços anteriores de (PINHEIRO, 2006) e (VIDAL, 2007), visou dar uma contribuição ao desenvolvimento de uma equipe de dirigíveis robóticos autônomos robustos e de baixo custo. Uma das formas conhecidas de se reduzir o custo de desenvolvimento de sistemas complexos é através da utilização de simulação computacional. Neste sentido, um software simulador de dirigíveis foi concebido. O software é capaz de

simular o comportamento físico do dirigível e tem a possibilidade de ser estendido com algoritmos inteligentes, fornecendo diversos subsídios para isso.

O simulador desenvolvido consiste em um software capaz de apresentar, em um cenário virtual tridimensional, dirigíveis navegando sobre terrenos de diversos tipos, contendo objetos de diferentes tipos que aparentam objetos do mundo real. Os dirigíveis podem ser construídos dentro do próprio software de simulação e diversos parâmetros podem ser configurados, de forma a deixar o dirigível com a maior semelhança visual e comportamental possível em relação aos dirigíveis reais. O ambiente de atuação do dirigível também pode ser construído pelo usuário, sendo possível simular ambientes abertos e fechados. Nos ambientes abertos, o terreno é formado por um mapa de relevo, que consiste em uma imagem monocromática, cujas dimensões determinam o tamanho do terreno simulado e a profundidade de cor do pixel determina a altitude do terreno no ponto relativo à posição do pixel na imagem. Os ambientes fechados consistem em um plano envolvido por quatro paredes e um teto, sendo todas as dimensões configuráveis. Em ambos os tipos de ambiente é possível inserir objetos tridimensionais, objetos estes que podem ser obtidos em algum repositório (como na internet) ou podem ser construídos em ferramentas de modelagem 3D.

Uma das características marcantes do simulador é que ele é capaz de reproduzir o comportamento físico dos dirigíveis. Isso foi obtido implementando-se o modelo matemático desse tipo de veículo diretamente no simulador, ou seja, sem a utilização de ferramentas externas, como o Matlab. As forças e os torques atuantes em cada dirigível são continuamente calculados durante a simulação, usando os conceitos da física newtoniana e o modelo matemático adaptado de (GOMES e RAMOS, 1998). Para a determinação contínua das variáveis de estado, o método de integração de Euler foi utilizado.

O desenvolvimento de algoritmos inteligentes, com a finalidade de proporcionar algum grau de autonomia aos dirigíveis, é uma questão crucial para o projeto. A extensão do simulador com esses algoritmos é outra funcionalidade por ele oferecida. Através de um protocolo de rede em nível de aplicação, é possível interagir com o simulador, que passa a assumir o papel de servidor de simulação. Um sistema cliente externo, remotamente conectado através de soquetes, pode obter informações dos sensores e comandar os atuadores dos dirigíveis durante uma simulação.

Foram realizados diversos testes de desempenho no simulador. Os testes serviram para analisar o desempenho de renderização do cenário tridimensional e da interface de comunicação via soquetes. As análises mostraram que o desempenho, mesmo em um computador de configuração modesta quando se trata de aplicações gráficas, foram satisfatórios, confirmando a viabilidade de se utilizar o simulador para o fim proposto. Uma observação importante com relação ao desempenho é que a renderização 3D de ambientes externos (*outdoors*) é mais custosa computacionalmente do que a de ambientes fechados (*indoors*).

Com o objetivo de validar e demonstrar a utilização remota do simulador e de realizar um estudo sobre o uso de técnicas de aprendizado de máquina para determinar a trajetória entre dois pontos, dois outros sistemas clientes foram desenvolvidos, sendo que um deles resolve o problema de determinação de caminhos usando o algoritmo clássico A* e o outro resolve o mesmo problema usando o algoritmo de aprendizado *Q-Learning*. Os resultados obtidos com o algoritmo *Q-Learning* foram satisfatórios e mostraram que seu uso deve ser considerado ao se desenvolver sistemas de determinação de caminhos. Muitas melhorias e abordagens distintas ainda podem ser feitas, tanto no simulador quanto nos sistemas de determinação de caminhos. A próxima subseção apresenta algumas sugestões para trabalhos futuros, visando a continuidade do projeto.

7.2 TRABALHOS FUTUROS

Apesar de apresentar resultados e contribuições relevantes para o problema geral já citado, o presente trabalho abre um leque para novos trabalhos, principalmente porque muitos pontos não puderam ser aprofundados, devido ao espaço e ao escopo deste trabalho. Além disso, com um simulador, o desenvolvimento de determinados trabalhos dispensam a utilização de dirigíveis reais. Algumas sugestões para trabalhos futuros são:

- Desenvolver um sistema de reconstrução tridimensional de ambientes, utilizando as imagens das câmeras virtuais embarcadas no dirigível simulado;

- Com a aquisição de dirigíveis reais, construir dirigíveis virtuais que se comportem como os reais, usando o mecanismo de construção de dirigíveis do simulador e, se necessário, ajustar o modelo matemático já implementado;

- Desenvolver um mecanismo de localização baseando-se nas discrepâncias do relevo do terreno simulado, usando algoritmos de *matching* ou análise de clusters para comparar amostragens de medições de altitude com pontos do mapa de relevo do ambiente (que deve ser conhecido), buscando identificar a posição do dirigível relativa ao SCA;

- Melhorar a qualidade da apresentação das imagens tridimensionais, usando recursos avançados suportados pelas placas gráficas mais recentes;

- Desenvolver e analisar com maior profundidade sistemas de planejamento de trajetória, sejam eles baseados em marcações artificiais no solo, em seguimento de marcas contínuas (como estradas e rios) etc.;

- Desenvolver sistemas de controle que sejam capazes de comandar os atuadores do dirigível para que ele siga uma trajetória pré-definida por um sistema de planejamento de trajetória externo;

- Aumentar a realidade do terreno simulado, através de um mecanismo que permita a construção de terrenos mesclados, como ilhas envoltas por mar, relevos com texturas variando de acordo com altura, colocando, por exemplo, uma textura de neve em pontos de altitude elevada e de areia em baixas altitudes;

- Desenvolver sistemas inteligentes diversos que sejam capazes de tirar proveito da utilização de vários dirigíveis e de planejar ações de forma que elas sejam executadas colaborativamente pelos dirigíveis;

- Implementar o modelo matemático diretamente na GPU para melhorar o desempenho da simulação;

- Utilizar APIs de física desenvolvidas por terceiros, principalmente as que abordam dinâmica de fluidos, visando aperfeiçoar a precisão e o realismo da simulação.

Outras idéias podem ser derivadas das supracitadas, considerando as tarefas intrínsecas presentes na execução de uma tarefa autônoma, como navegação, controle e planejamento de trajetória.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AZEVEDO, E., CONCI, A. **Computação Gráfica: Teoria e Prática**. 4ª Edição. Editora Campus. Rio de Janeiro, 2003. ISBN 85-351-1253-3.

BARAFF, D. **An Introduction to Physically Based Modeling: Rigid Body Simulation I – Unconstrained Rigid Body Dynamics**. Carnegie Mellon University, Robotics Insititute. Carnegie Mellon, 1997.

BELLMAN, R. E. **Dynamic Programming**. Princeton University Press, Princeton, New Jersey, 1957. ISBN-13 978-0-486-42809-3.

BELLMAN, R. **On a Routing Problem**. Quarterly of Applied Mathematics, 16(1), pp.87-90, 1958.

BOIFFIER, J. L. **The Dynamics of Flight: The Equations**. Wiley. Chichester, England, 1998. ISBN-13 978-0-471-94237-5.

BORENSTEIN, J., EVERETT, H. R., FENG, L. **Where am I? Sensors and Methods for Mobile Robot Positioning**. Technical Report, The University of Michigan. Michigan, 1996.

BOTELHO, W. T. **Um Sistema de Identificação e Adaptação Pervasivo para a Casa Inteligente Utilizando Sistemas Multiagentes**. Dissertação de Mestrado. Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2005.

BOURG, D. M. **Physics for Game Developers**. O'Reilly. Sebastopon, 2002. ISBN-13 978-0-596-00006-6.

BOURG, D. M., SEEMANN, G. **AI for Game Developers**. O'Reilly. Sebastopon, 2004. ISBN-13 978-0-596-00555-9.

BRUTZMAN, D. **Virtual World Visualization for an Autonomous Underwater Vehicle**. Proceedings of the IEEE Oceanic Engineering Society Conference OCEANS 95, pages 1592-1600, San Diego, CA, October 1995.

CHELLI, E., DALA, L., et al. **Aerodynamics Investigation of the 1/80 CL 160 P1 Airship**. 14th AIAA Lighter-Than-Air Technical Conference, (14: jul. 2001:Akron-EUA). Proceedings AIAA, Akron, USA, 2001.

CONGER, D. **Physics Modeling for Game Programmers**. Course PTR. Boston, MA, 2004. ISBN-13 978-1-592-00093-7.

CRAIG, J. J. **Introduction to Robotics: Mechanics and Control**. Addison Wesley Longman. Massachusets, 1989. ISBN-13 978-0-201-54361-2.

DE LAURIER, J. D., EVANS, J. R. **The Shenandoah Flies Again: A Computer Simulation**. 4th AIAA Lighter-Than-Air Technology Conference, Annapolis, Maryland, USA, 1981.

DELOACH, S. A. **Analysis and Design Using MaSE and AgentTool**. Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio, March 31, Abril 2001ª.

DELOACH, S. A., WOOD, M. F. **Developing Multiagent Systems with Agent-Tool**. Proceedings of the Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2001b.

DIJKSTRA, E. W. **A note on two problems in connexion with graphs**. Numerische Mathematik. 1, S. 269–271, 1959.

DUCHAINEAU, M. A., WOLINSKY, M., SIGETI, D. E., MILLER, M. C., ALDRICH, C., WEINSTEIN, M. B. M. **ROAMing terrain: Real-time optimally adapting meshes**. IEEE Visualization, pages 81–88, 1997.

EBERLY, D. H. **Game Physics**. Morgan Kaufmann. San Francisco, CA, 2004. ISBN-13 978-1-558-60740-8.

ELFES, A., BUENO, S. S., BERGERMAN, MARCEL, RAMOS, J. J. G. **A Semi-Autonomous Robotic Airship for Environmental Monitoring Missions**. Proccedings of the 1998 IEEE. International Conference on Robotics & Automation. Leuven, Belgium. May 1998.

FARIA, B. G. **Identificação Dinâmica Longitudinal de um Dirigível Robótico Autônomo**. Dissertação de Mestrado. Universidade Estadual de Campinas, Campinas, Brasil, 2005.

FERBER, J., GASSER, L. **Intelligence Artificielle Distribuée**. International Workshop on Expert Systems & Their Applications. Avignon. Cours n. 9. France: [s.n], 1991.

FILHO, C. P. **Introdução à Simulação de Sistemas.** Editora da Unicamp. Campinas, 1995. ISBN 85-268-0349-2.

FLOYD, R. W. **Algorithm 97: Shortest path**. Commun. ACM 5, 6 (Jun. 1962), 345. New York, NY, USA, 1962.

GOMES, S. B. V. **An investigation of the flight dynamics of airships with application to the YEZ-2A**. Ph.D. Thesis, College of Aeronautics, Cranfield University, England, 1990.

GOMES, S. B. V., RAMOS, J. J. G. **Airship Dynamic Modeling for Autonomous Operation**. Proccedings of the 1998 IEEE. International Conference on Robotics & Automation. Leuven, Belgium. May 1998.

HART, P. E., NILSSON, N. J., RAPHAEL, B. **A Formal Basis for the Heuristic Determination of Minimum Cost Paths**. IEEE Transactions on Systems Science and Cybernetics SSC4 (2): pp. 100–107, 1968.

HIBELER, R. C. **Dinâmica: Mecânica para Engenharia**. 10ª Edição. Pearson Prentice Hall. São Paulo, 2005. ISBN 85-879-1896-6.

HIBELER, R. C. **Estática: Mecânica para Engenharia**. 10ª Edição. Pearson Prentice Hall. São Paulo, 2005. ISBN 85-879-1897-4.

HOUAISS, A., VILLAR, M. S., FRANCO, F. M. M. **Dicionário Houaiss da Língua Portuguesa**. 1ª Reimpressão. Editora Objetiva. Rio de Janeiro, 2004. ISBN 85-7302-383-X.

HOUAISS, A., VILLAR, M. S., FRANCO, F. M. M. **Dicionário Houaiss: Sinônimos e Antônimos**. 1ª Edição. Editora Objetiva. Rio de Janeiro, 2003. ISBN 85-7302-487-9.

LIU, J., WU, J. **Multiagent Robotic Systems**. CRC Press. Washington, D.C., 2001. ISBN-13 978-0-849-32288-4.

KAELBLING, L. P., LITTMAN, M. L., MOORE, A. W. **Reinforcement Learning: A Survey**. Journal of Artificial Intelligence Research 4 : 237-285, 1996.

KHOURY, G. A., GILLETT, J. D. **Airship Technology**. Cambridge University Press. Cambridge, UK, 1999. ISBN-13 978-0-521-60753-7.

KILGARD, M. J. **The OpenGL Utility Toolkit (GLUT) Programming Interface**. Silicon Graphics Inc., 1996.

KODICEK, D. **Mathematics and Physics for Programmers**. Charles Rivers Media. Hingham, 2005. ISBN-13 978-1-584-50330-9.

KRÖPLIN, I. B. **Solar Airship LOTTE**. University of Stuttgart. Disponível: http://www.isd.uni-stuttgart.de/arbeitsgruppen/airship/infodownload/engl_v02.pdf, capturado em 14 de abril de 2007.

KRUS, P. **Basic Numerical Integration Methods for Simulation**. Linköping University, Department of Mechanical Engineering. Disponível: http://www.machine.ikp.liu.se/edu/post/multidomain/basicnumericalintegrationmethods.pdf , capturado em 5 de abril de 2007.

LATOMBE, J. C. **Robot Motion Planning**. Kluwe Academic Publishers, Boston, 1991. ISBN-13 978-0-792-39129-6.

LAUMOND, J. P. **Robot Motion Planning and Control**. Springer. New York, 1998. ISBN-13 978-3-540-76219-5.

MCCORMICK, B. W. **Aerodynamics, Aeronautics and Flight Mechanics**. Wiley. 1995. ISBN-13 978-0-471-11087-3.

MCREYNOLDS, T., BLYTHE, D. **Advanced Graphics Programming Using OpenGL**. Morgan Kaufmann Publishers. San Francisco, CA, 2005. ISBN-13 978-1-558-60659-3.

MENEZES, P. J. C. **Estudos em Navegação de Robôs Móveis**. Dissertação de Mestrado, Universidade de Coimbra, 1999.

MOLLER, A. T., HAINES, E. **Real-Time Rendering**. AK Peters, 2nd Edition, 2002. ISBN-13 978-1-568-81182-6.

NICOLA, J., INFANTE, U. **Gramática Contemporânea da Língua Portuguesa**. 13ª Edição. Editora Scipione. São Paulo, 1994. ISBN 85-262-1397-0.

OPENGL Open Graphics Library. Online. Disponível: http://opengl.org, capturado em 3 de julho de 2007.

PALMER, G. **Physics for Game Programmers**. Apress. Berkeley, CA, 2005. ISBN-13 978-1-590-59472-8.

PINHEIRO, C. A. P. **Veículos Aéreos Autônomos Não-Tripulados para Monitoramento de Ambientes Desestruturados e Comunicação de Dados**. Dissertação de Mestrado. Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2006.

POLACK, T., LAMOTHE, A. **Focus on 3D Terrain Programming**. Premier Press. Cincinnati, Ohio, 2002. ISBN-13 978-1-592-00028-9.

RAMOS, J. J. G. **Contribuição ao Desenvolvimento de Dirigíveis Robóticos**. Tese de Doutorado. Universidade Federal de Santa Catarina. Florianópolis, Março de 2002.

RAMOS, J. J. G., BRUCIAPAGLIA, A., BUENO, S. **An Internet Based Airship Simulator**. Technical Report LRV-1997-13, Automation Institute-CTI.

RAMOS, J. J. G., MAETA, S. M., MIRISOLA, L. G. B., BERGERMAN, M., BUENO, S. S., PAVANI, G. S., BRUCIAPAGIA, A. **A Software Environment for an Autonomous Unmanned Airship**. Proceedings of the 1999 IEEE/ASME. International Conference on Advanced Intelligent Mechatronics. September 19-23. Atlanta, USA.

RAMOS, J. J. G., PAIVA, E. C., BUENO, S. S., MAETA, S. M., MIRISOLA, L. G. B., BERGMAN, M., FARIA, B. G. **Autonomous Flight Experiment With a Robotic Unmanned Airship**. Proceedings of the 2001 IEEE. Seoul, Korea. May 21-26, 2001 International Conference on Robotics & Automation.

REIS, L. P. **Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico**, Capítulos 3 e 5. Tese de Doutoramento, Faculdade de Engenharia de Universidade do Porto – FEUP, Julho 2003.

REZENDE, S. O. **Sistemas Inteligentes: Fundamentos e Aplicações**. Manole. Barueri, SP, 2005. ISBN 85-204-1683-7.

RÖETTGER, S., HEIDRICH, W., SLUSALLEK, P., SEIDEL H. P. **Real-Time Generation of Continuous Levels of Detail for Height Fields**. Proceedings of WSCG '98, pages 315-322, 1998.

ROSA, P. F. F., VIDAL, F. S., BERNARDO, R. M., PINHEIRO, C. A. P., PELLANDA, P. C. **A Low Cost UAV System for Data Communication, Surveillance and Monitoring Using Heterogeneous Blimps**. In: XII Latin-American Congress on Automatic Control, 2006, Salvador. Proceedings of the XII Latin-American Congress on Automatic Control, 2006. p. 366-370.

ROSKAM, J. **Airplane Flight Dynamics and Automatic Flight Controls**. DARcorporation. Lawrence, KS, 2001. ISBN-13 978-1-884-88518-1.

RUSSELL, S.,NORVIG, P. **Inteligência Artificial**. 2ª Edição. Editora Campus. Rio de Janeiro, 2004. ISBN: 85-352-1177-2.

SAEEDI, P., LAWRENCE, P. D., Lowe, D. G. **Vision-Based 3-D Trajectory Tracking for Unknown Environments**. Proceedings of the IEEE Transactions on Robotics, Vol 22, no. 1, pp 119-136, 2006.

SCHMIDT, L. V. **Introduction to Aircraft Flight Dynamics**. AIAA Education Series. Reston, Virginia, 1998. ISBN-13 978-1-563-47226-8.

SEGAL, M., AKELEY, K. **The OpenGL Graphics System: A Specification**. Silicon Graphics, Inc., 2006. ASIN B0006RHZAA.

SIEGWART, R., NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. The MIT Press. Cambridge, Massachusetts, 2004. ISBN-13 978-0-262-19502-7.

SINOPOLI, B., MICHELI,M., DONATO, G., KOO, T. J. **Vision Based Navigation for an Unmanned Aerial Vehicle**. Proceedings of the IEEE International Conference on Robotics an Automation. Seoul, Korea. May, 2001.

SOUSA, J., SIMSEK, T., VARAIYA, P. **Task Planning and Execution for UAV Teams**. IEEE Conference on Decision and Control, pp 3804-3810. Atlantis, Paradise Island, Bahamas, 2004.

SUTTON, R., BARTO, A. **Reinforcement Learning: An Introduction**. Mit Press, Cambridge, MA, 1998. ISBN-13 978-0-262-19398-6.

SWOKOWSKI, E. W. **Cálculo com Geometria Analítica**. Vol. 1. 2ª Edição. Makron Books. São Paulo, 1994. ISBN 85-346-0308-1.

TALAY, T. A. **Introduction to the Aerodynamics of Flight**. NASA. Washington, D.C., 1975. ASIN B0006CMZD2.

TIPLER, P. A., MOSCA, G. **Física**. Vol. 1. 5ª Edição. LTC. Rio de Janeiro, 2006. ISBN 85-216-1462-4.

UPSON, R., KLIKOFF, W. **Application of Practical Hydrodynamics to Airship Design**. British Aeronautical Research Commitee Report and Memoranda, No 405, 1925.

VARAIYA, P. **Hierarchical Control of Semi-Autonomous Teams Under Uncertainty**. University of California, Final Report. California, USA, 2004.

VELHO, L, GOMES, J. **Sistemas Gráficos 3D**. Instituto de Matemática Pura e Aplicada - IMPA. Série de Computação e Matemática. Rio de Janeiro, 2001. ISBN 85-244-0167-2.

VICCI, L. **Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation**. Technical Report. University of North Carolina. April 2001.

VIDAL, F. **Sistema de Navegação para Dirigíveis Aéreos Não-Tripulados Baseado em Imagens**. Dissertação de Mestrado. Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2007.

WATKINS, C. J. C. H. **Learning with Delayed Rewards**. PhD Thesis, Cambridge University Psychology Department, 1989.

WATKINS, C. J. C. H., DAYAN, P. **Q-Learning**. Machine Learning, 8 (3/4): 279-292, 1989.

WIKIPEDIA History of Unmanned Aerial Vehicles. Online. Disponível: http://en.wikipedia.org/wiki/History_of_unmanned_aerial_vehicles, capturado em 11 de maio de 2007.

WILLIAMS, S. B., DISSANAYAKE, G., DURRANT-WHYTE, H. **An Efficient Approach to the Simultaneous Localization and Mapping Problem**. Proceedings of the 2002 IEEE. International Conference on Robotics & Automation. Washington, USA. May 2002.

WOOD, M. F., DELOCACH, S. A. **An Overview of The Multiagent Systems Engineering Methodology**. Lecture Notes in Computer Science. Vol. 1957, Springer Verlag, Berlin, January 2001.

ZAMSTEIN, L. M., ARROYO, A. A., SCHWARTZ, E. M. **Koolio: Path Planning Using Reinforcement Learning on a Real Robot Platform**. Florida Conference on Recent Advances in Robotics. Miami, Florida, May 25-26, 2006.

9 APÊNDICE

9.1 APÊNDICE 1: MANUAL DE UTILIZAÇÃO DO SIMULADOR

Para utilizar o simulador desenvolvido no presente trabalho, doravante denominado "SimuBlimp", é necessário conhecer suas funcionalidades. As subseções deste anexo apresentam todas as opções de criação e configuração de ambientes e de dirigíveis presentes no SimuBlimp.

9.1.1 A TELA PRINCIPAL

A FIG. 9.1 mostra a tela principal do simulador. Nesta figura, é possível notar que, na barra de ferramentas, apenas o primeiro, o sexto e o sétimo botões estão habilitados. O sexto botão mostra uma janela de ajuda com os comandos do simulador. O sétimo botão executa a saída do sistema. O primeiro botão aciona a criação do ambiente de simulação. Ao clicar nesse botão, que é o único botão de controle habilitado, a janela da FIG. 9.2 é exibida.

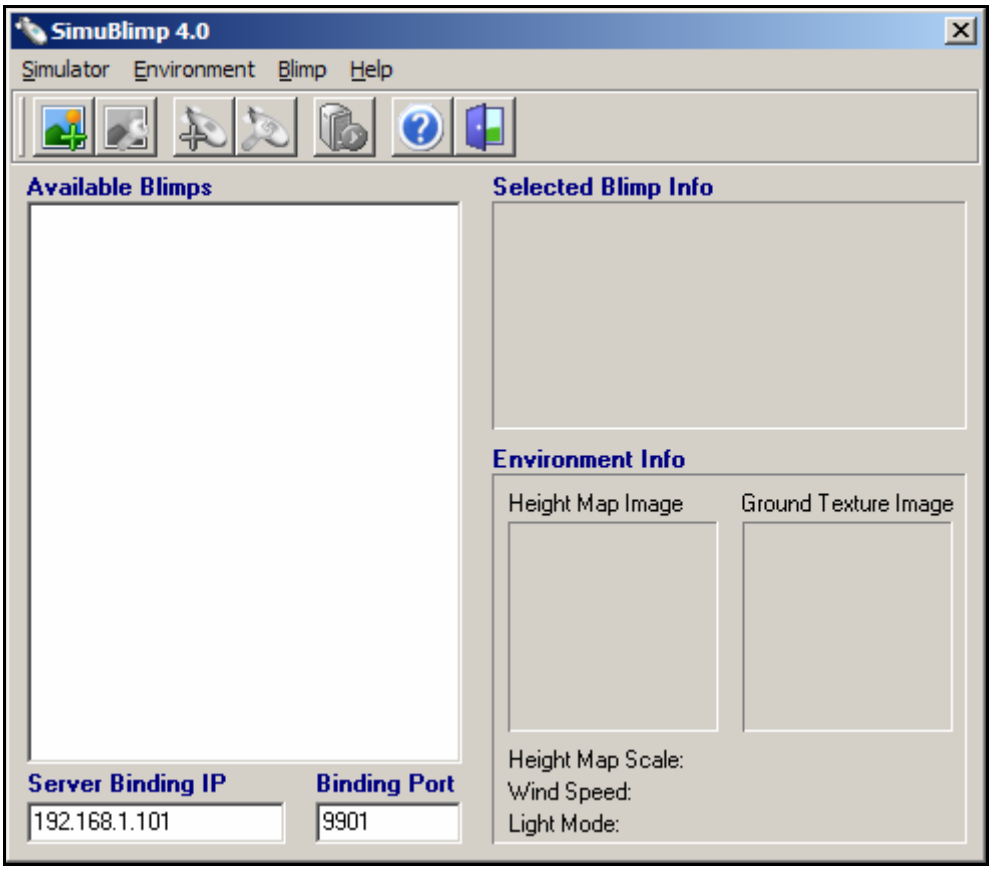

FIG. 9.1 – Tela principal do SimuBlimp.

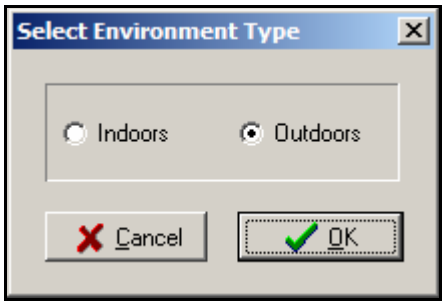

FIG. 9.2 – Janela de seleção de tipo de ambiente.

Na FIG. 9.2, ao clicar no botão OK, se a opção "*Indoors*" estiver marcada, a janela da FIG. 9.3 será mostrada e, nessa janela, um ambiente fechado poderá ser configurado. Se a opção "*Outdoors*" estiver marcada, a janela da FIG. 9.4 será mostrada, sendo que nessa janela um ambiente aberto poderá ser configurado. As subseções seguintes apresentam maiores detalhes sobre as janelas de criação de ambientes.

9.1.2 CRIANDO UM AMBIENTE FECHADO

Se na janela da FIG. 9.2 a opção "*Indoors*" estiver marcada, ao clicar no botão OK a janela da FIG. 9.3 será exibida. Essa janela permite criar um ambiente fechado, sendo que o usuário deve fornecer as características do ambiente, como as dimensões, as texturas e os objetos que estarão presentes no ambiente. A FIG. 9.3 mostra a janela de construção de ambientes fechados, cujos elementos são apresentados a seguir. A listagem "*Avaiable Objects*", na parte superior esquerda da janela, apresenta uma listagem com objetos pré-configurados que podem ser inseridos diretamente no ambiente, clicando no botão com rótulo ">". Clicando no botão "+", é possível importar um objeto de um arquivo externo, ou seja, que não esteja na listagem, desde que o arquivo tenha a extensão 3DS ou OBJ. Os botões "S" e "L" servem, respectivamente, para salvar uma lista de objetos já inseridos e para carregar uma lista de objetos salva em outra ocasião. Os objetos já inseridos no ambiente passam a aparecer na listagem "*Environment Objects*". Nessa listagem, uma caixa de checagem aparece do lado esquerdo de cada item. Se a caixa de checagem estiver marcada, o objeto estará visível no ambiente, caso contrário, o objeto não será mostrado. A FIG. 9.3 mostra que 4 objetos já foram inseridos no ambiente que está sendo criado e que todos eles estão visíveis. Cada objeto pode ter algumas propriedades individuais modificadas, como a posição, a escala e a rotação do objeto sobre o eixo Z. Para modificar as propriedades de um objeto, basta selecioná-lo na lista de objetos já inseridos e configurar os respectivos campos no canto superior direito da janela.

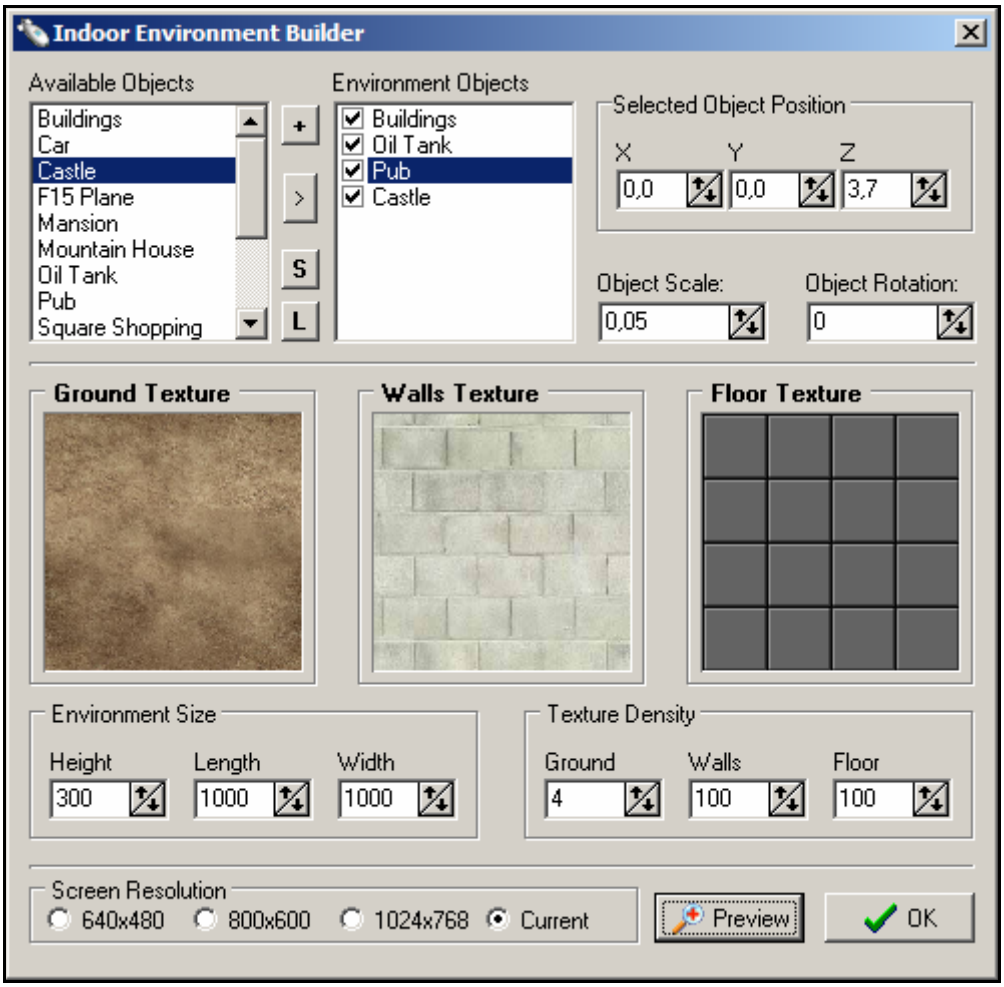

FIG. 9.3 – Janela de criação de ambientes fechados (*indoors*).

As três figuras no meio da janela da FIG. 9.3 permitem configurar, respectivamente, a textura do chão, a textura das paredes e a textura do teto de um ambiente fechado. Para alterar a textura, basta clicar sobre a figura para selecionar uma nova imagem, que deve ser uma imagem com extensão JPG ou BMP. Na região nomeada "*Environment Size*", é possível configurar a altura, o comprimento e a largura do ambiente fechado, nessa ordem. Na região "*Texture Density*", pode-se configurar a densidade da textura aplicada no chão, nas paredes e no teto, respectivamente. Quanto menor for o valor informado, menos densa será a textura. O botão "*Preview*" permite pré-visualizar o ambiente recém criado, antes mesmo de iniciar a simulação. A resolução de exibição da pré-visualização pode ser configurada à esquerda do botão, sendo que, nesse caso, o ambiente sempre é mostrado em tela cheia e na resolução selecionada. Após fornecer todas as configurações desejadas para a criação do ambiente, basta clicar no botão OK para retornar à tela principal e prosseguir com a criação dos dirigíveis para, posteriormente, iniciar a simulação.

9.1.3 CRIANDO UM AMBIENTE ABERTO

Se na janela da FIG. 9.2 a opção "*Outdoors*" estiver marcada, ao clicar no botão OK a janela da FIG. 9.4 será exibida. Essa janela permite criar um ambiente aberto, sendo que, assim como é feito para ambientes fechados, o usuário deve fornecer algumas características do ambiente, as quais diferem um pouco das características de um ambiente fechado. As dimensões do terreno, a textura do terreno e os objetos que estarão presentes no ambiente são algumas das características configuráveis, como é elucidado mais adiante. Um mapa de relevo do ambiente também é necessário. A FIG. 9.4 mostra a janela de criação de ambientes abertos.

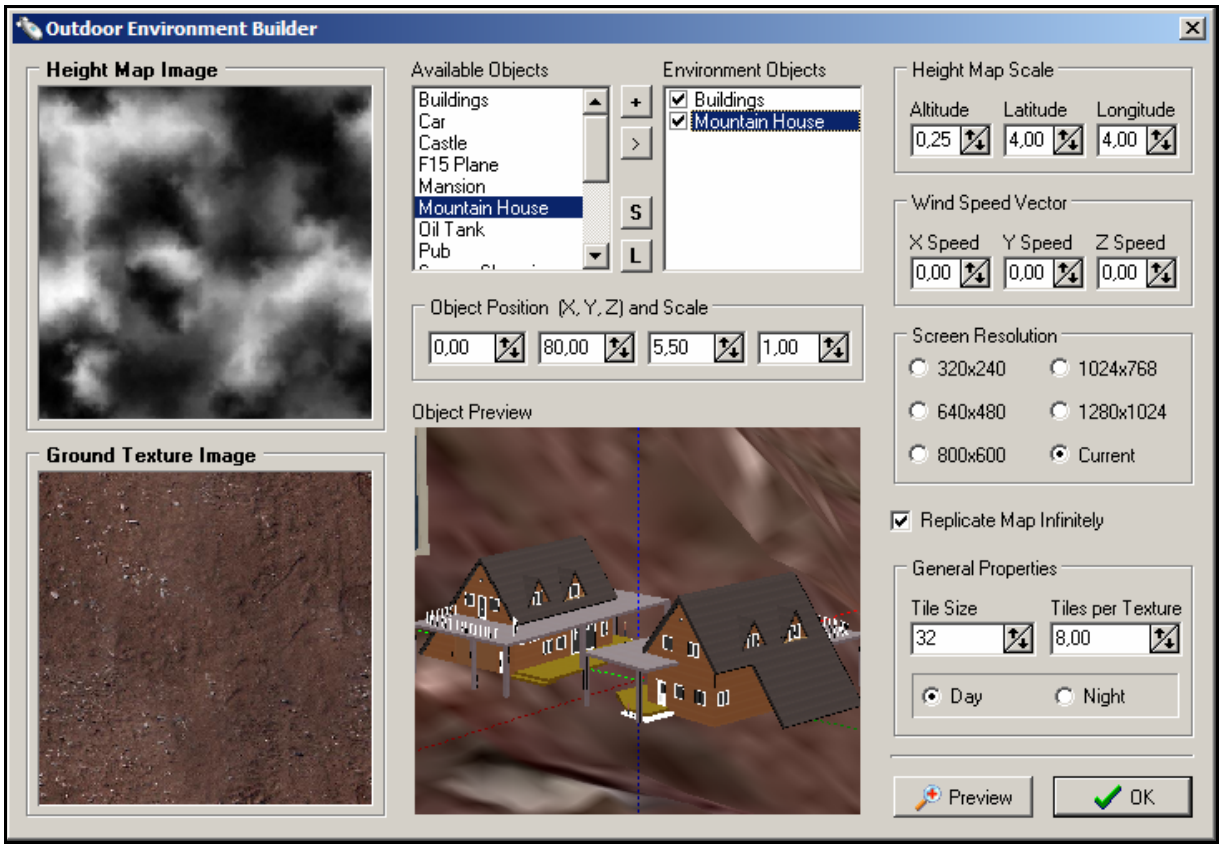

FIG. 9.4 – Janela de criação de ambientes abertos (*outdoors*).

Na FIG. 9.4, a imagem do canto superior esquerdo, na região "*Height Map Image*", é o mapa de relevo. Esse mapa de relevo é responsável por determinar as variações de altitude do terreno. Para selecionar outro mapa de relevo, deve-se clicar sobre a figura e selecionar uma imagem com extensão JPG ou BMP que represente um mapa de relevo. Na imagem logo abaixo, na região "*Ground Texture Image*", é possível configurar a textura do terreno. Para

alterar a textura, basta clicar sobre a figura e selecionar uma nova imagem JPG ou BMP que contenha à textura desejada.

As dimensões do terreno são determinadas pelo tamanho do mapa de relevo e pelos fatores de escala. Na região "*Height Map Scale*" é possível configurar esses fatores de escala. No caso da largura e do comprimento do terreno, os campos "*Latitude*" e "*Longitude*" determinam o fator de escala. Por exemplo, se esses valores estiverem ambos configurados como 4, cada pixel no mapa de relevo corresponderá a uma área de 4 x 4 metros.

O fator de escala de altitude é definido no campo "*Altitude*". No mapa de relevo, a altitude de um ponto no terreno é determinada pelo valor de qualquer um dos componentes de cor do pixel na respectiva coordenada, pois, sendo a imagem monocromática, todos os componentes RGB do pixel têm o mesmo valor. Por exemplo, se o pixel de coordenada (32, 128) tem um componente de cor igual a 200 e o fator de escala é 0,25, o terreno terá uma altitude de 50 metros na respectiva coordenada.

Outra opção que altera a configuração do terreno em si é a caixa de checagem "*Replicate Map Infinitely*", que, quando marcada, replica o mapa lado a lado infinitamente. Se a opção estiver desmarcada, apenas uma área determinada pelo tamanho do mapa de relevo e pelos fatores de escala será renderizada, como ocorre na FIG. 5.10. Na região "*General Properties*", é possível determinar a densidade da textura no campo "*Tile Size*", sendo que, quanto menor for o valor, mais densa será a textura. Ainda, é possível determinar se o céu exibido no cenário será um céu diurno ou noturno, deixando marcada, respectivamente, a opção "*Day*" ou "*Night*".

A inserção de objetos no ambiente aberto é feita de forma análoga à inserção de objetos em ambientes fechados. A única diferença é que, na inserção de objetos em ambientes abertos, há uma pré-visualização do objeto inserido na própria tela de criação, tendo em vista que, devido às variações de altitude do terreno, quase sempre a posição do objeto no eixo Z deverá ser modificada, para que o objeto fique exatamente sobre o terreno. Na área de visualização do objeto, é possível clicar com o botão esquerdo do mouse sobre ele e rotacionar a câmera, movendo o mouse. Para aproximar e afastar a câmera, basta clicar com o botão direito na imagem e mover o mouse para frente e para trás.

Na região "*Wind Vector Speed*", pode-se configurar a velocidade e a direção do vento presente no ambiente. A velocidade é determinada pela magnitude do vetor composto pelos três valores e a direção é composta por esse mesmo vetor, normalizado. Assim como na criação de ambientes fechados, após configurar um ambiente aberto, é possível pré-visualizar o terreno, bastando clicar no botão "*Preview*". Após a configuração do ambiente, deve-se clicar no botão OK para voltar à tela principal do simulador e prosseguir com a definição dos dirigíveis da simulação.

Caso algum detalhe do ambiente (aberto ou fechado) tenha sido esquecido, é possível acessar novamente a janela de criação do ambiente e alterar a configuração desejada. Essa tarefa pode ser realizada clicando-se no segundo botão da barra de ferramentas da tela principal, mostrado na FIG. 9.5 (b), que se torna habilitado após a criação do ambiente. Contudo, não é permitida a mudança de tipo para um ambiente já criado, ou seja, alternar entre ambiente aberto e fechado. Caso isso seja necessário, deve-se reiniciar o simulador e reconstruir o ambiente desde o início.

FIG. 9.5 – Botões da barra de ferramentas da janela principal.

9.1.4 CRIANDO OS DIRIGÍVEIS

Após a criação do ambiente, o terceiro botão da barra de ferramentas da janela principal, mostrado na FIG. 9.5 (c), passa a estar habilitado. Isso quer dizer que agora é possível inserir dirigíveis no ambiente. Clicando nesse botão, antes de mostrar a tela de criação de dirigíveis, é apresentada uma janela onde se deve determinar um nome para o dirigível que está sendo criado. A FIG. 9.6 mostra essa janela.

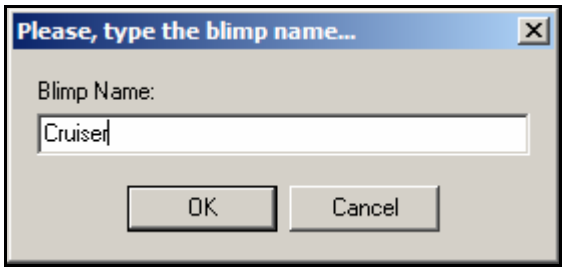

FIG. 9.6 – Janela de solicitação do nome do dirigível a ser criado.

A subseção 5.4 apresenta detalhes sobre a criação de cada um dos componentes do dirigível e as configurações possíveis de serem feitas em cada tipo de componente. A FIG. 9.7 mostra a barra de ferramentas da janela de criação de dirigíveis.

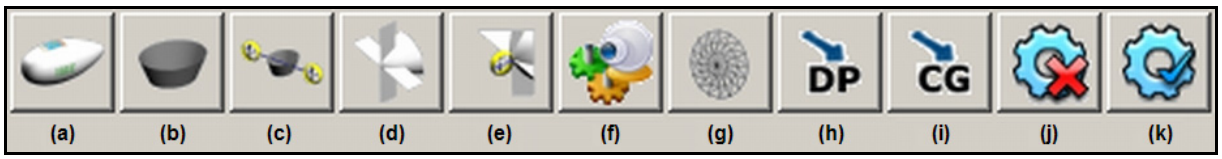

FIG. 9.7 – Botões da barra de ferramentas da oficina virtual.

Os botões da FIG. 9.7 possuem as seguintes funcionalidades:

- (a) – Este botão é específico para a criação do envelope de gás do dirigível. Este é o primeiro componente que deve ser criado. Ao clicar no botão, a janela da FIG. 5.13 é mostrada;

- (b) – Este botão acessa a janela de criação da gôndola, mostrada na FIG. 5.15. A gôndola só pode ser criada após o envelope;

- (c) – Este botão acessa a janela de criação dos propulsores de gôndola, que só podem ser criados após a gôndola. A FIG. 5.17 mostra a janela de criação dos propulsores de gôndola;

- (d) – A janela de criação dos lemes de cauda é acessada através desse botão. Os lemes de cauda só podem ser criados após os propulsores de gôndola. A FIG. 5.19 mostra a janela de criação dos lemes de cauda;

- (e) – A criação do propulsor de cauda é feita através da janela da FIG. 5.21, acessada através desse botão. A criação do propulsor de cauda deve ser posterior à criação dos lemes de cauda;

- (f) – Este botão acessa a janela de configuração de equipamentos embarcados, mostrada na FIG. 5.23. Os equipamentos embarcados só podem ser configurados após a criação da gôndola;

- (g) – Este botão alterna os modos de exibição do dirigível que está sendo construído entre: texturizado, polígonos e pontos;

- (h) – Este botão restaura a posição dos componentes do dirigível para a posição determinada durante a criação de cada um dos componentes componente;

- (i) – Este botão calcula o centro de gravidade do dirigível e reajusta as coordenadas dos componentes, de forma que o centro de gravidade passe a ser a origem do SCL;

- (j) – Este botão cancela a criação de um dirigível, retornando à janela principal. Não está disponível quando a oficina é acessada para alterar um dirigível já criado;

- (k) – Este botão confirma a criação do dirigível que está sendo configurado e retorna à tela principal, adicionando o novo dirigível à lista "*Available Blimps*".

Uma observação importante é que, sempre que se terminar a configuração dos componentes de um dirigível na oficina de criação de dirigíveis, é necessário clicar no botão da FIG. 5.7 (i) para determinar o centro de gravidade do dirigível. Caso contrário, ao iniciar a simulação, os torques que deveriam atuar no dirigível não poderão ser determinados, pois o centro de gravidade seria desconhecido.

Os dirigíveis criados podem ser alterados, clicando-se no quarto botão da barra de ferramentas, mostrado na FIG. 9.7 (d). Esse botão só estará habilitado se um dirigível estiver selecionado na listagem "*Available Blimps*", que mostra todos os dirigíveis já criados. Nessa mesma listagem, é possível ativar ou desativar um dirigível, clicando na caixa de checagem à esquerda do nome do dirigível que se deseja ativar ou desativar. Se a caixa de checagem estiver marcada, o dirigível estará presente na simulação. Se estiver desmarcada, o dirigível continuará criado, mas não estará presente na simulação.

9.1.5 INICIANDO A SIMULAÇÃO

Após criar um ambiente e um ou mais dirigíveis para atuar nesse ambiente, é possível dar início à simulação. Contudo, antes de dar início à simulação, deve-se configurar o IP e a porta que o servidor de simulação fará uso para aceitar acessos remotos. O IP deve ser um IP válido que localize o computador que está executando o simulador em alguma rede TCP/IP. Na tela principal, mostrada na FIG. 9.1, o IP pode ser configurado no campo "*Server Binding IP*" e a porta usada pelo servidor, no campo "*Binding Port*". Feito isso, basta clicar no botão da FIG. 9.5 (e) para iniciar a simulação. Esse botão só estará disponível se o ambiente e pelo menos um dirigível já estiverem criados.

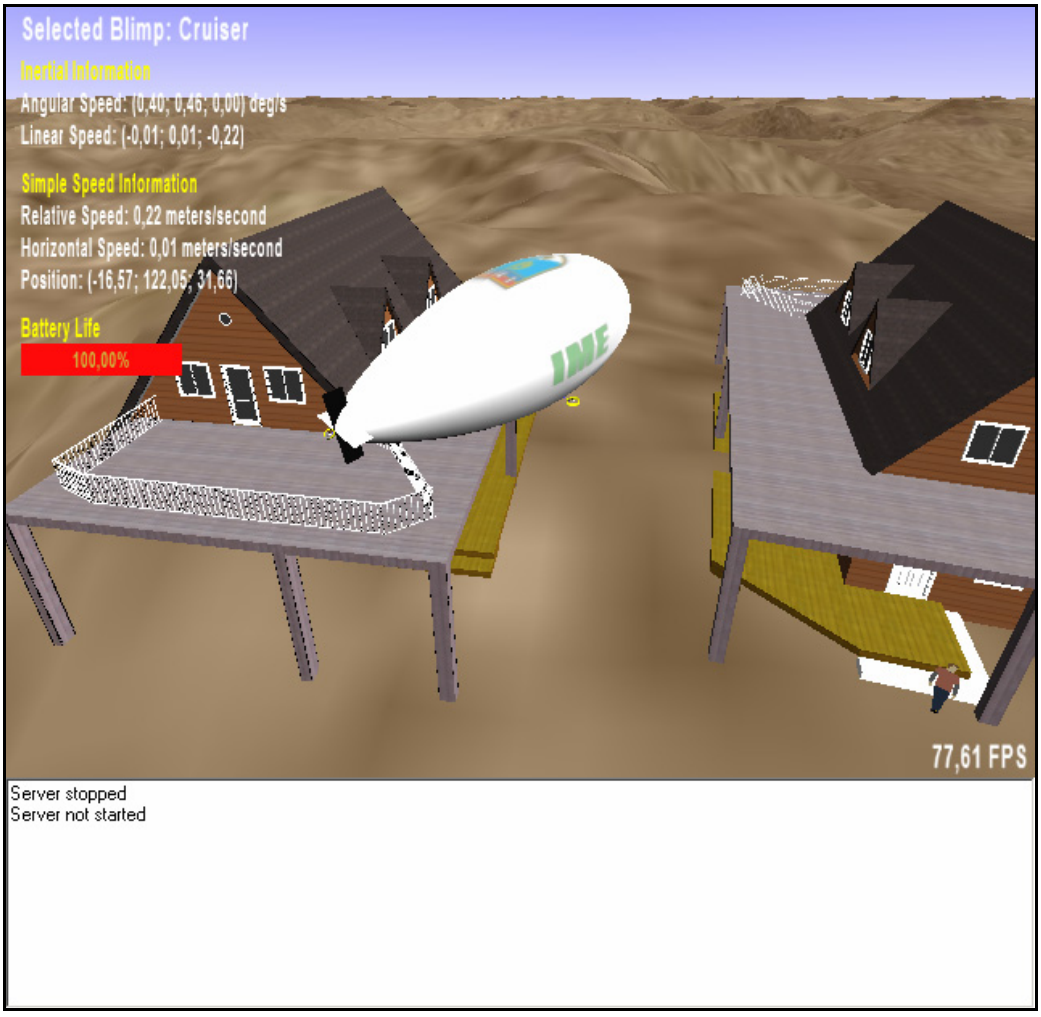

FIG. 9.8 – Simulação em execução.

Assim que é iniciada a simulação, um cenário 3D com o ambiente e os dirigíveis é exibido. Logo abaixo do cenário, é exibida uma tela de status do servidor de simulação, que

apresenta mensagens relacionadas aos eventos que ocorrem entre o servidor e algum cliente remoto conectado. A FIG. 9.8 mostra uma simulação em execução. Nessa figura, é possível observar que algumas informações são exibidas no canto superior direito do cenário. Essas informações são relativas ao dirigível selecionado em um dado instante, que, nesse caso, é o dirigível de nome "*Cruiser*". Essas mesmas informações podem ser obtidas remotamente por um cliente conectado ao simulador.

Alguns comandos podem ser executados via teclado durante uma simulação. Esses comandos permitem que o usuário acione os atuadores e sensores dos dirigíveis, permitindo que o usuário observe o comportamento e as reações de um dirigível de acordo com o comando dado. A TAB. 9.1 mostra os comandos de teclado que podem ser usados durante a simulação.

TAB. 9.1 – Comandos de teclado disponíveis durante a simulação.

Tecla	Função
F3	Seleciona a câmera embarcada esquerda do dirigível selecionado (estéreo)
F4	Seleciona a câmera embarcada direita do dirigível selecionado (estéreo)
F5	Seleciona a câmera embarcada central do dirigível selecionado (monocular)
F6	Seleciona a câmera externa do dirigível selecionado
F7	Seleciona a câmera do ambiente
Esc	Termina a simulação
Home	Seleciona o dirigível anterior
End	Seleciona o próximo dirigível
PageUp	Aumenta a altitude da câmera (apenas câmera do F6 e do F7)
PageDown	Reduz a altitude da câmera (apenas câmera do F6 e do F7)
Seta Acima	Aumenta a potência dos propulsores da gôndola para frente
Seta Abaixo	Aumenta a potência dos propulsores da gôndola para trás
Seta à Esquerda	Aumenta a potência do propulsor de cauda para a esquerda
Seta à Direita	Aumenta a potência do propulsor de cauda para a direita
G	Desliga os propulsores da gôndola
C	Desliga o propulsor da cauda
A	Rotaciona o eixo dos propulsores de gôndola para cima quando forem vetorizáveis
Z	Rotaciona o eixo dos propulsores de gôndola para baixo quando forem vetorizáveis
I	Exibe/oculta informações em tela sobre o dirigível selecionado
1	Altera o tamanho do cenário para 320 x 240 pixels
2	Altera o tamanho do cenário para 640 x 480 pixels
3	Altera o tamanho do cenário para 800 x 600 pixels
4	Altera o tamanho do cenário para 1024 x768 pixels
5	Altera o tamanho do cenário para 1280 x 1024 pixels

Além dos comandos de teclado da TAB. 9.1, também estão disponíveis alguns comandos através do mouse. Mantendo o botão esquerdo do mouse pressionado sobre o cenário, ao mover o mouse, a câmera rotaciona sobre o objeto selecionado, mas somente se a câmera selecionada for a do ambiente (acionada pela tecla F7) ou a externa ao dirigível (acionada pela tecla F6). Se o botão direito do mouse for mantido pressionado sobre o cenário, caso o mouse

seja movimentado para frente e para trás, a câmera se aproxima e se afasta do objeto focalizado. Os controles para manipulação local do simulador têm o objetivo de proporcionar ao usuário um aprendizado sobre o comportamento dinâmico do dirigível e como os comandos interferem em sua navegação e configuração. Como já mencionado, o uso principal proposto para o simulador é o de servir como fonte de dados a sistemas clientes que desejam interagir com o simulador para executar algoritmos inteligentes capazes de realizar alguma tarefa intrínseca ao problema geral.